SPEZIFIKATION

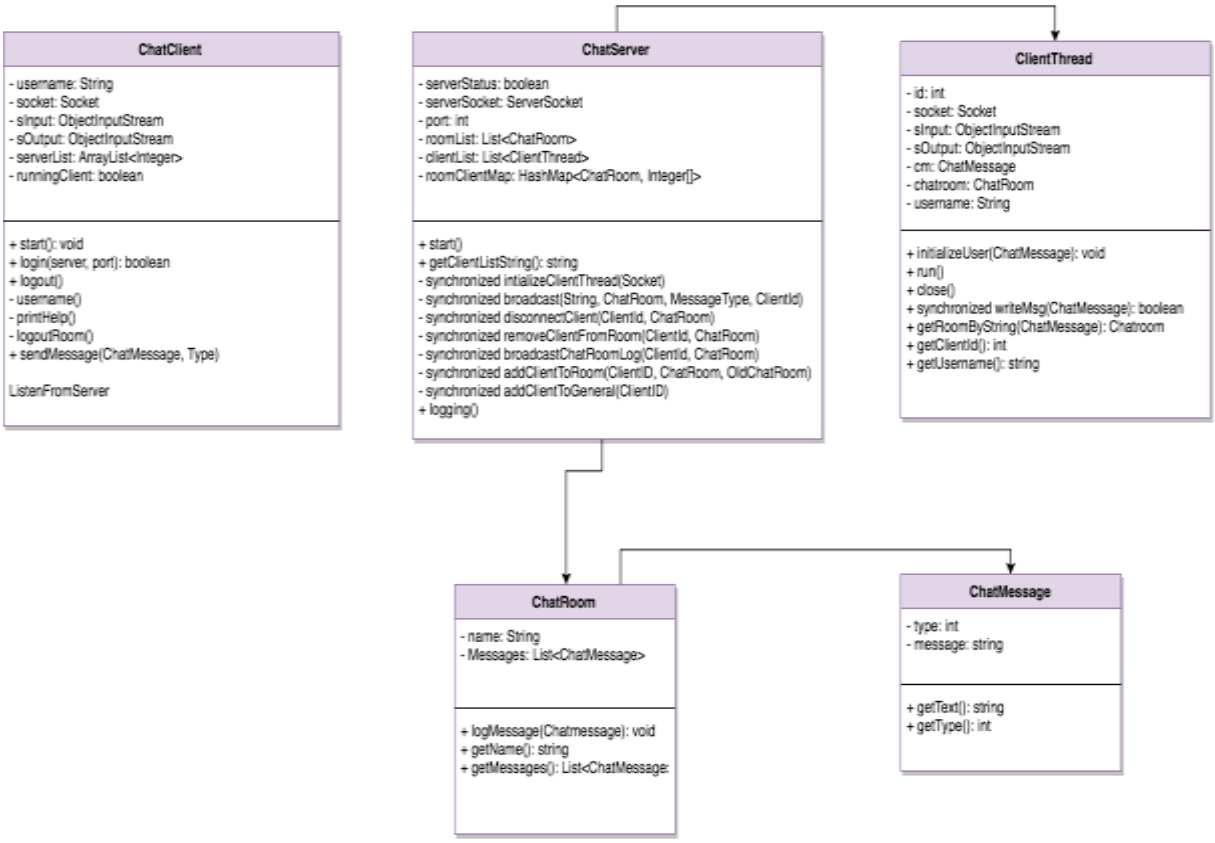
Amanpreet Kaur & Till Poland

MAY 22, 2017

RECHNERNETZE PRAKTIKUM – AUFGABEN 2 & 3

Inhaltsverzeichnis

1. Systemüberblick



1.1 ChatClient

Die Komponente ChatClient kann sich als Client mit dem Server unterhalten. Von ihm aus ist es dem Benutzer möglich, sich am Server und darauf bei Chatrooms anzumelden und mit anderen Clients zu chatten.

Außerdem kann sich der Client über Befehle mit dem Server unterhalten. Gesendet werden können ChatMessages, und nur diese werden auch sinngemäß vom Server verstanden. Der Typ der ChatMessage ist dabei mit einem bestimmten Ereignis verbunden.

1.2 ChatServer

Der ChatServer erstellt eine Socketverbindung mit einem entsprechenden Port und lauscht auf einkommende Verbindungen. Sobald sich ein Client mit dem ChatServer verbindet wird ein ClientThread für den Client angelegt. Über den ClientThread kann der Server sich mit dem Client unterhalten, sprich er kann Nachrichten empfangen und versenden. Der Client wird nach der Verbindung automatisch in den so genannten “General”-Room (Eingangs-Zimmer) hineingelassen. Damit ist der Client beim Server angemeldet und der Handshake um sich mit dem Server zu verbinden und zu unterhalten ist erledigt.

1.3 ClientThread

Wie im ChatServer bereits beschrieben, ist das die “offene” Verbindung zum Client. Der ClientThread wird erstellt, sobald sich ein ChatClient beim Server anmeldet. Der ClientThread ist mit einer eindeutigen Nummer versehen und muss von dem Client mit einem Username benannt werden. Jeder ClientThread wird zudem einem Chatroom zugeordnet, wobei sich jeder Client nur in einem Chatroom befinden kann.

Der ClientThread erledigt nun alles auf dem Server, was den jeweiligen Client anbelangt. Und über das Socket kann er mit dem Client kommunizieren. Also kann der Client über diesen Thread eine Nachricht den Server schicken und den Server zu weiteren Handlung auffordern.

Sobald eine Nachricht vom Client eingeht wird anhand des NachrichtenTyps entschieden wie die Nachricht behandelt wird. Es kann sich z.B. um eine INITIALIZE, JOIN_CHATROOM usw. Nachrichten handeln. (Siehe Technische Details - Befehle)

1.4 ChatRoom

Der ChatRoom hat einen eindeutigen Namen, über den dieser angesprochen bzw. erreicht werden kann. In einem Chatroom können sich mehrere Clients unterhalten, sprich sie verschicken Nachrichten untereinander. Jeder Chatroom verwaltet eine Nachrichtenliste, mit den Nachrichten die in diesem Raum versendet wurden. Sprich der Raum führt ein Protokoll über die gesendeten Nachrichten und gibt dieses auch an neue Benutzer im Raum an.

1.5 ChatMessage

Der Server ist so implementiert, dass er nur Nachrichten vom Typ ChatMessage akzeptiert. Die ChatMessage hat eine Nachricht (String) und einen Typ (Integer) als Parameter. Die ChatMessages sind über einen Typ mit einem bestimmten Ereignis verbunden. Wenn der Client eine Nachricht an den Server bzw. Chatroom verschicken möchte muss diese Nachricht vom Typ ChatMessage<String, Int> sein.

2. Technische Aspekte

Unter diesem Punkt werden die technischen Aspekte des Servers und Clients aufgelistet.

2.1 Server

Der Server ist ein auf Java 1.8 basierender Server, der die Clients als einzelne Threads verwaltet und die Aufrufe von zu synchronisierenden Methoden über den Java Monitor reguliert. So kann sichergestellt werden, dass keine Funktionen durch gleichzeitigen Zugriff auf Ressourcen Probleme bereiten können.

Er hört kontinuierlich auf Socketanfragen auf seine IP und seinen Port.

2.2 Port

Der von uns benutzte Port ist der Port 3000.

2.3 Protokoll

Vom Client aus werden ChatMessage Objekte über einen Socketstream über eine TCP Verbindung gesendet, welche in den technischen Details spezifiziert sind.

2.3.1 Anmeldung von Chatclient

Zur Anmeldung am Server baut der Client eine Verbindung mit dem Server über ein Socket auf. Die Socketverbindung verläuft über die IP des gestarteten Servers (zu Testzwecken z.B. localhost) und über den oben spezifizierten Port.

a. Darauf folgend, muss eine Initialisierungsnachricht gesendet werden, die folgendermaßen aufgebaut ist:

   Text: Der gewählte Benutzername

   Typ: Initialize

   Info: Zwei unterschiedliche Benutzer können den gleichen Benutzernamen haben!

b. Weitere Kommunikation

   Nach der Initialisierung nimmt der Server alle der unter “Technische Details” spezifizierten Befehle entgegen und beantwortet diese sinngemäß. Dieser Austausch findet über dieselbe Socketverbindung statt, mit der auch die Initialisierung stattgefunden hat.

   Wird diese Verbindung unterbrochen, muss sich der Client auf dem Server neu initialisieren.

2.3.2 Timeout

Der Benutzer wird nach 5 Minuten, in der er/sie nichts geschrieben hat, automatisch abgemeldet und vom Server gelöscht.

2.3.3 Fehlermeldungen

Fehlermeldungen werden als Strings ausgegeben, behindern aber nicht den weiteren Verlauf des Chats. Sie sind bei den Befehlen in den Technischen Details aufgelistet.

2.3.4 Textkodierung

Die Nachrichten müssen in Unicode (Utf-8) kodiert sein, da dies der nativen Stringkodierung von Java entspricht. Andernfalls kann der Server diese gegebenenfalls nicht korrekt darstellen bzw initialisieren.

3. Technische Details

In dem Folgend Abschnitt werden die Technischen Daten unseres Systems beschrieben und zusammengefasst.

3.1 SendeObjekt

Das Objekt, das vom Server akzeptiert wird, ist einzig ein ChatMessage Objekt. Dieses Objekt muss die im UML im Systemüberblick genannten Funktionen unterstützen und außerdem Serializable sein, da sie über einen Java ObjectOutputStream an den Client versendet und über einen ObjectInputStream empfangen wird.

3.2 Chatrooms

Falls sich der letzte Benutzer aus einem Chatroom ausloggt, wird der Chatroom direkt gelöscht.

3.3 Befehle

Die Befehle, die auf dem Server ausgeführt werden, werden über einen Integer Wert typisiert. Je nachdem, welcher Integer gesendet wird, wird die gesendete Nachricht anders vom Server behandelt.

Die Antwort vom Server ist immer ein String. Auch wenn es eine Fehlermeldung ist.

Die Liste der Typen sieht folgendermaßen aus:

Logout - Wert: 0

Folge:	Abmeldung und Löschung von dem aufrufenden Benutzer auf dem Server.
Antwort d. Servers:	Keine Antwort.

Message - Wert: 1

Folge:	Die Nachricht wird an alle anderen Benutzer im momentanen ChatRoom gesendet, außer der Benutzer ist im generellen ChatRoom.
Antwort d. Servers:	Keine Antwort.

List Users - Wert: 2

Folge:	Der Server gibt eine Liste aller momentan angemeldeten Benutzer heraus, aufgelistet mit usernamen.
Antwort d. Servers:	Liste aller angemeldeten Benutzer

Join Chatroom - Wert: 3

Folge:	Der Benutzer wird aufgefordert, den Namen des beizutretenden ChatRooms einzugeben.
Antwort d. Servers:	Log des ChatRooms
Fehler:	ChatRoom nicht vorhanden

List Chatrooms – Wert 4

Folge: Der Server listet alle ChatRooms auf, die auf dem Server existieren

Antwort d. Servers: Liste aller ChatRooms, die auf dem Server gespeichert sind.

Chatroom logout- Wert 5

Folge: Der Benutzer wird aus dem momentanen Chatroom gelöscht und in den generellen ChatRoom verschoben.

Falls der Benutzer schon im generellen Chatroom war, passiert nichts.

Antwort d. Servers: Keine Antwort

Create Chatroom – Wert 6

Folge:

1. Von dem Benutzer wird ein Name abgefragt (Antwort d. Servers), der als Name für den Server benutzt werden soll.
2. Nach der Eingabe eines Namens wird der ChatRoom mit dem gewählten Namen erstellt und der Benutzer dem ChatRoom hinzugefügt.

Antwort d. Servers: Keine Antwort

In Chatroom – Wert 8

Folge: Der Server gibt den Namen des Chatrooms zurück, in dem sich der Benutzer momentan befindet.

Antwort d. Servers: Momentaner Chatroom

Users in chatroom – Wert 9

Folge: Der Server gibt als String eine Liste aller Benutzer zurück, die momentan in dem Chatroom sind, in dem der Benutzer sich befindet.

Antwort. d. Servers: Liste aller Benutzer, die in dem Chatroom sind, in dem der aufrufende Benutzer sich befindet.

Initialize – Wert 11

Folge: Wird zur Initialisierung am Server verwendet. Der Benutzer ist mit dem mitgegebenen Benutzernamen initialisiert.

Antwort d. Servers: Keine Antwort