**RNP Entwurf Chat-System**

**Autoren: Timo Peters, Agin Punneliparambil**

Inhaltsverzeichnis

[Einleitung 2](#_Toc499136908)

[Spezifikation/Implementierung 2](#_Toc499136909)

[Sequenzdiagramm 4](#_Toc499136910)

[Protokoll 5](#_Toc499136911)

[Allgemein 5](#_Toc499136912)

[Verbindungsaufbau 6](#_Toc499136913)

[Anmeldung 6](#_Toc499136914)

[Lobby 6](#_Toc499136915)

[Chatraum 6](#_Toc499136916)

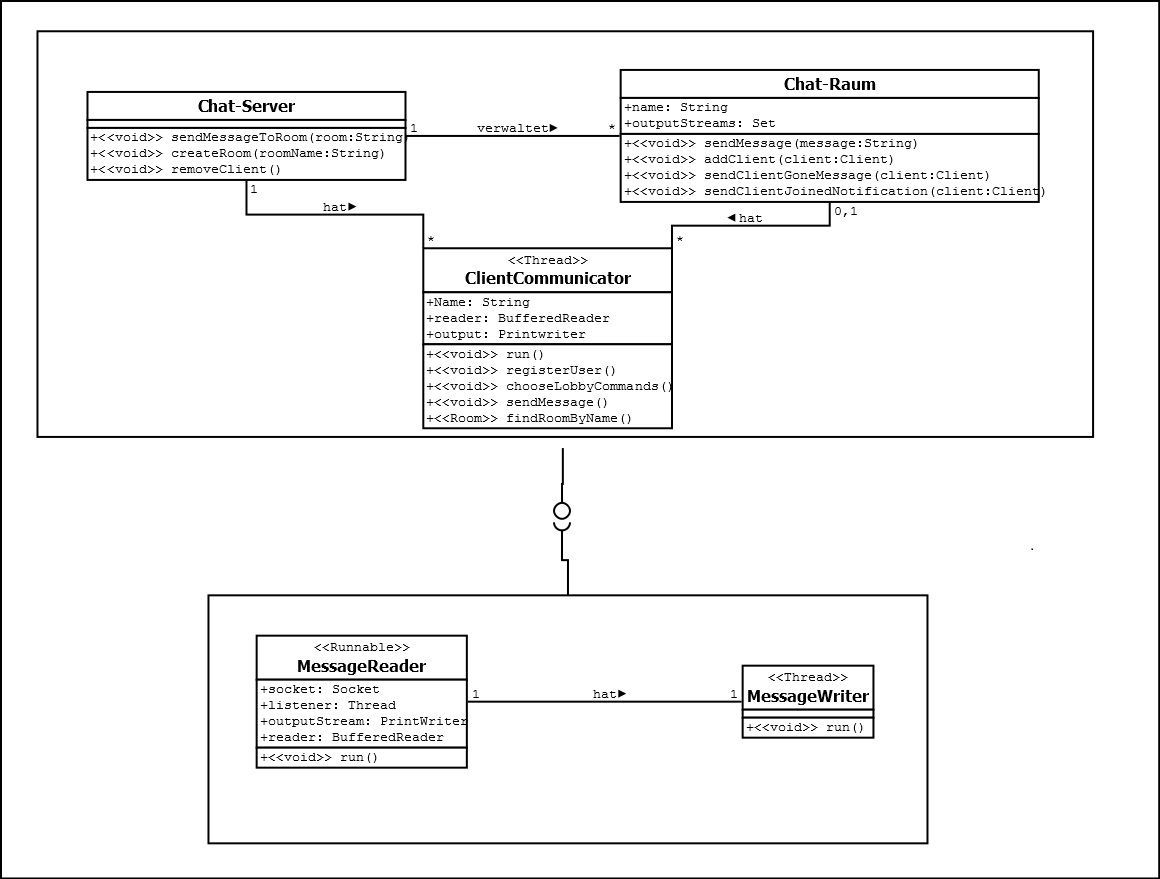
# Einleitung

In diesem Entwurf wird die Umsetzung eines Chat-Systems erläutert. Zu einem Chat-System gehören zwei wesentliche Komponenten, der Chat-Client, der die Benutzerinteraktion realisiert und ein Chat-Server, der Chat-Räume verwaltet und für die Nachrichtenverteilung verantwortlich ist.

Der Server soll ein Socket anbieten um einen Verbindungsaufbau mit einem Client zu ermöglichen. Die Kommunikation erfolgt über TCP. Der User kann über die Kommandozeile Nachrichten verschicken.

# Spezifikation/Implementierung

**UML-Klassendiagramm**



Das Chat-System besteht aus 2 Komponenten: ServerComponent und ClientComponent. ServerComponent verwaltet einen Server und stellt über einen TCP-Socket jederzeit eine Verbindung zur ClientComponent bereit.

Die ServerComponent besteht aus den drei Klassen: Server, Room und ClientCommunicator.

**ChatServer:**

Die Klasse ChatServer stellt den Socket zur erstmaligen Verbindung eines Clients. Außerdem besitzt sie die Listen rooms und names. Dadurch hat sie die Aufgabe der Raumverwaltung und durch die indirekten Referenzen auf die Clients auch eine Sammlung aller Clients, die mit dem Server verbunden sind. Außerdem enthält sie die main Methode zum Starten eines Servers.

**ChatRoom:**

ChatRoom repräsentiert einen Chatraum. Sie enthält: Einen Namen „name“, eine Referenz auf das zugehörige ChatServer-Objekt, eine Liste mit zugehörigen ClientCommunicator-Objekten und ein Set mit den OutputStreams der ClientCommunicator-Objekte. Ihre Hauptaufgabe ist die Verteilung von Messages an alle ClientCommunicator-Objekte, die zu dem ChatRoom gehören. Die Messages kommen aus der ClientComponent über die ClientCommunicator an den ChatServer und schlussendlich an den ChatRoom, welcher die Message über die OutputStream-Liste broadcastet.

**ClientCommunication:**

ClientCommunicator wird als Thread umgesetzt. Ein Objekt dieser Klasse wird dann initialisiert, wenn die ClientComponent auf den Socket des ChatServer-Objektes zugreift. Diese Klasse stellt der ClientComponent einen Verbindungssocket für die Kommunikation zur Verfügung.

In der run () Methode wird dazu das Protokoll umgesetzt.

Die ClientComponent besteht aus zwei Klassen: MessageReader und MessageWriter.

Diese sind beide als Thread umgesetzt und teilen sich die Aufgabe der Kommunikation mit der ServerComponent. Dies ist essentiell, um das Nachrichten schreiben und empfangen zu synchronisieren.

**MessageReader:**

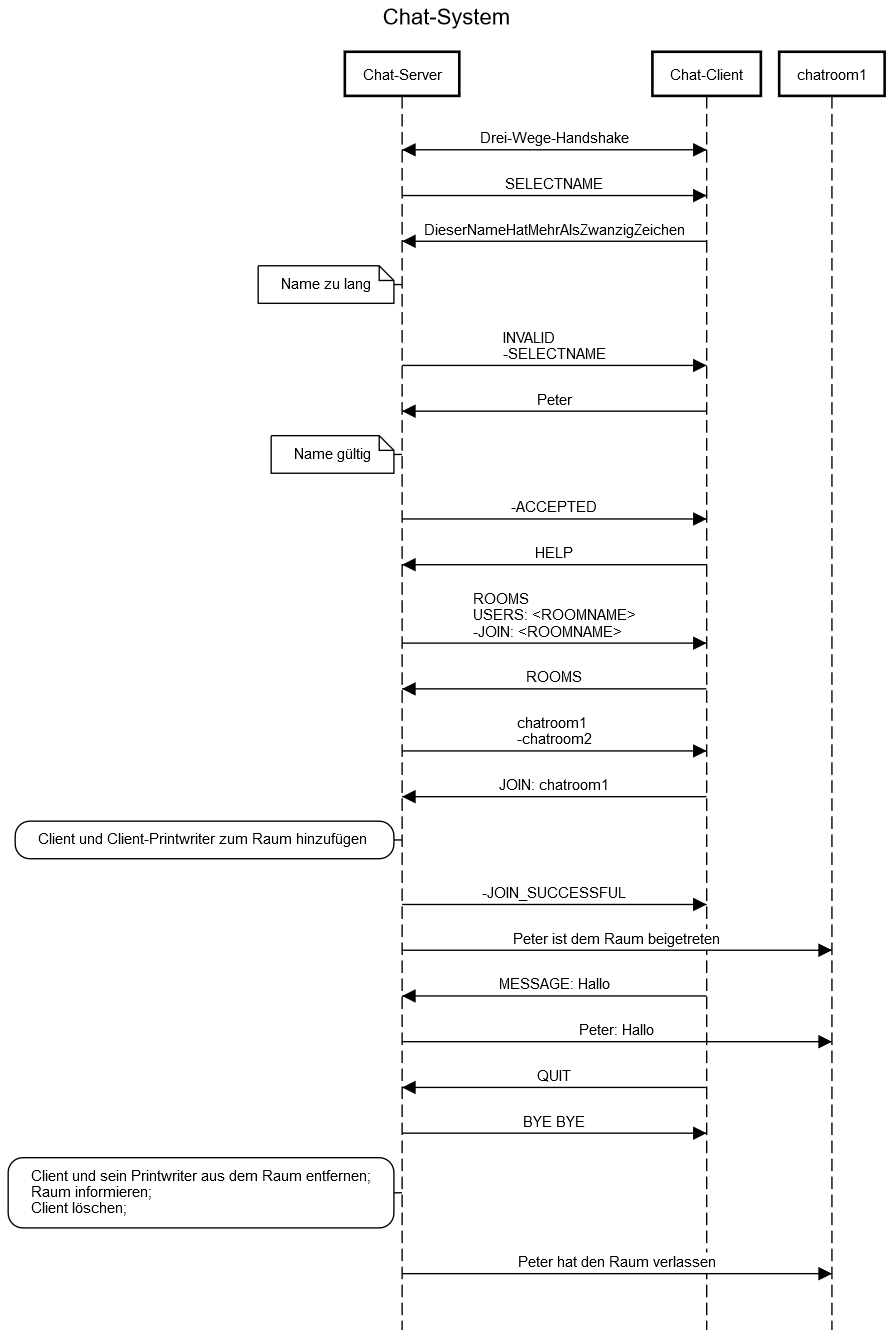
MessageReader hat 2 Hauptaufgaben: Kommunikation und Verbindung mit der ServerComponent. Sie verfügt über die main Methode, über die ein Client gestartet wird. In der main wird ein Socket und dessen in- und outputstream initialisiert. Außerdem wird ein neuer MessageWriter Thread gestartet. In der run Methode werden bis zum Verbindungsverlust zum Server permanent Nachrichten aus dem OutputStream abgefragt.

**MessageWriter:**

MessageWriter hat die Aufgabe inputs an die ServerComponent zu senden. Dafür erhält ein Objekt dieser Klasse eine Referenz auf ein MessageWriter-Objekt und dessen zugehörigen Socket. Die Klasse nutzt den InputStream des MessageWriters um jederzeit einen input an die ServerComponent zu senden.

# Sequenzdiagramm

Dieses Sequenzdiagramm zeigt die Interaktion der zwei Komponenten. Obwohl der „chatroom1“ keine Komponente ist, wird er hier zum besseren Verständnis als Akteur dargestellt.



# Protokoll

## Allgemein

Der Server gibt nach jeder Eingabe (außer beim Nachrichtenversand im Chatraum) eine Rückmeldung. Die Serverrückmeldung kann aus einer oder mehreren Zeilen bestehen. Die letzte Zeile einer Serverrückmeldung erkennt man anhand des Präfixes „- “.

**Beispiel:**

***ROOMS***

***USERS: <ROOMNAME>***

***-JOIN: <ROOMNAME>***

Folgende Tabelle enthält alle Befehle, die vom Server erkannt werden. In der Spalte „zulässiger Abschnitt“, ist die erlaubte Benutzung definiert. Ein Aufruf der Befehle in einem ungültigen Abschnitt wird keine Funktion nach sich ziehen und der Server gibt die Rückmeldung:

***-UNKNOWN\_COMMAND***

Oder

***-INVALID\_MESSAGE\_FORMAT***

|  |  |  |  |
| --- | --- | --- | --- |
| **Befehl** | **Beschreibung** | **Serverantwort** | **Zulässiger Abschnitt** |
| QUIT | Verbindungsabbruch | -BYE BYE | Anmeldung, Lobby, Chat |
| HELP | Zeigt Liste mit Lobby-Befehlen | ROOMS  USERS: <ROOMNAME>  -JOIN: <ROOMNAME> | Lobby |
| ROOMS | Zeigt die Liste mit allen Chaträumen | *Beispiel, wenn Raum vorhanden:*  Chatroom1  -Chatroom2  *Kein Raum vorhanden:*  - | Lobby |
| USERS: <roomname> | Zeigt alle Clients in einem bestimmten Raum <roomname>. Zwischen „:“ und dem Raumnamen muss sich genau ein Leerzeichen befinden. | *Beispiel, wenn Personen vorhanden:*  Bernd  -Peter  *Keine Person vorhanden:*  - | Lobby |
| JOIN: <roomname> | Beitritt zu einem Chatraum <roomname>. Zwischen „:“ und dem Raumnamen muss sich genau ein Leerzeichen befinden. | -JOIN\_SUCCESSFUL | Lobby |
| MESSAGE: <message> | Es wird eine Nachricht <message> an alle Clients im aktuellen Raum gesendet. Zwischen „:“ und der message muss sich genau ein Leerzeichen befinden. |  | Chatraum |

## Verbindungsaufbau

Server-IP: abhängig vom Host

Portnummer: abhängig von der Implementierung des Servers

Der Verbindungsaufbau erfolgt über TCP. Die dafür relevanten Informationen sind Server-IP und Portnummer mit denen der Client versucht eine Verbindung aufzubauen.

## Anmeldung

Bei einem erfolgreichen Verbindungsaufbau kommt vom Server die Antwort:

*-SELECTNAME*

Der Client wird hiermit aufgefordert einen Chatnamen zu wählen. Die Eingabe des Chatnamens muss dem Server nicht signalisiert werden. Der eingegebene Text wird auf Gültigkeit überprüft. Namen ohne Zeichen, im Präfix oder Suffix ein „Leerzeichen“ haben oder die eine Länge von 20 Zeichen überschreiten gelten als ungültig. Ebenso führt ein Duplikat (d.h. ein anderer Benutzer hat schon den Namen) dazu, dass der Name vom Server abgelehnt wird. Die entsprechende Rückmeldung ist:

*INVALID*

*-SELECTNAME*

Der Client wird aufgefordert erneut einen Namen einzugeben. Bei einem gültigen Namen ist die Rückmeldung:

*-ACCEPTED*

## Lobby

In die Lobby gelangt man nach einer erfolgreichen Anmeldung. Die Lobby ist der zentrale Abschnitt. Hier kann sich der Client alle Chaträume und ihre Clients anzeigen lassen oder einem Chatraum beitreten. Die dazugehörigen Befehle sind in der obigen Tabelle zu finden.

## Chatraum

Nachdem man einem Chatraum beigetreten ist, erhalten zunächst alle anderen Clients in diesem Raum eine Benachrichtigung über den Beitritt. Während sich ein Client in einem Chatraum befindet kann er Nachrichten an alle Clients des Raumes senden und empfangen. Verlässt ein Client einen Chatraum, so werden alle anderen Clients dieses Raumes eine Benachrichtigung über das Verlassen.

Der Chatraum ist der einzige Abschnitt in dem die Meldung:

***-INVALID\_MESSAGE\_FORMAT***

Vom Server gesendet werden kann. Dies kommt vor, wenn eine Eingabe erfolgt, die von MESSAGE: <message> oder QUIT abweicht.