**Dokumentation zur Client/Server-Übung in TP**

Die Implementierung meiner Übung besteht aus 2 verschiedenen Projekt. Es gibt die Serveranwendung und die Clientanwendung. Beide kommunizieren über sockets und den ObjectInput - bzw. ObjectOutputStreams aus der standard-java-bibliothek miteinander. Als Protokoll dienen hierbei Unterklassen der abstrakten Klasse “Message”. Message erhält lediglich ein Stringattribut welches den Typ und damit den Zweck des Protokolls definiert.

Somit kann ich mir die jeweils benötigten Protokolle zusammenbasteln und sie über den Socket hin und herschicken, vorrausgesetzt sie erben von der Klasse Message.

Nun folgen die jeweiligen Aufgaben. Gefordert wurde die Angabe des Kommunikationsprotokolls sowie eine Beschreibung des Kommunikationsverhalten zwischen Client und Server über ein Zustandsdiagramm.

**Aufgabe 1)**

Berechnungsserver

Für meinen Berechnungsserver habe ich die abstrakte Klasse “Message” 2 x konkret umgesetzt. Außerdem implementiert sie das Interface Serializable, damit es möglich ist ihre konkreten Klassen durch Streams zu schicken.

Einmal gibt es die Message-Klasse als Protokoll vom Client zum Server(“CalcMsg”) und das zweite mal als Protokoll vom Server zum Client(“ResultMsg”). “CalcMsg” enthält die Berechnung welche an den Server geschickt wird. “ResultMsg” enthält das Ergebnis der Berechnung welches dann wieder zurück an den Client geschickt wird.

Das Zustandsdiagramm für die Protokoll sieht als wie folgt aus:

**Client Server**

CalcMsg ------------------>

<-------------------ResultMsg

Beide Anwendungen fangen alle mir bisher aufgefallenen Exceptions ab und behandeln sie dementsprechend. Außerdem ist der Server in der Lage, mehrere Clients gleichzeitig aufzunehmen und sich um sie zu kümmern. Bei Beendigung des Servers oder eines Clients werden alle Ressourcen wieder freigegeben.

**Aufgabe 2)**

Authentifizierung

Passwort: 1234

Mein Server ist nun auch mit einer Authentifizierungsfunktion ausgestattet. Es ist nun notwendig sich über ein Passwort zu authentifizieren um die Serverdienste nutzen zu können.

Als Protokoll wurde die Klasse “AuthMsg” von der Klasse “Message” konkretisiert. Sie enthält ein zusätzliches String-Attribut, welches das zu versendende Passwort enthält. Kommt das Protokoll am Server an, so wird das Passwort “entpackt” und mit einem im Quellcode festgelegten Passwort (Standard: 1234) verglichen. Stimmen die Passwörter überein, so fährt das Programm wie gewohnt fort. Ist dies nicht der Fall, so wird solange zur erneuten Passworteingabe aufgefordert bis die Passwörter übereinstimmen.

Zustandsdiagramm bei erfolgreicher Passworteingabe:

**Client Server**

Authmsg(**PW**) ------------------>

<-------------------ResultMsg(**PW**)

Zustandsdiagramm bei der Eingabe von einem falschen Passwort:

**Client Server**

Authmsg(**PW**) ------------------>

<-------------------ResultMsg(**null**)

...(Wird solange fortgeführt bis das Passwort richtig ist)

Authmsg(**PW**) ------------------>

<-------------------ResultMsg(**PW**)