Die Aufgaben können (müssen aber nicht) in Gruppen von bis zu drei Leuten bearbeitet werden. Falls die Aufgaben in einer Gruppe bearbeitet werden, genügt **eine** Abgabe mit den Namen aller Gruppenmitglieder. Abgabe bis **spätestens Freitag 23.07.2010** per Mail mit dem Betreff **PMSS2010 HA11 <Matrikelnummer>** (z. B. „PMSS2010 HA11 12345678“) an **pm@cs.uni-kassel.de**. Für diese Hausaufgabe gibt es 37 Punkte.

Hinweis zur Abgabe: Die Abgabe **MUSS** als exportiertes Eclipse Projekt erfolgen. Ist das Projekt nicht korrekt exportiert, kann es bei der Korrektur nicht berücksichtigt werden (es bietet sich also an, den Import des exportierten Projekts auszuprobieren).

In dieser Hausaufgabe soll ein einfacher Chat entwickelt werden, um die Grundlagen der Client/Server Programmierung zu erlangen. Hierzu wird zunächst der Server erstellt, danach der Client. Im Anschluss soll für den Client eine grafische Oberfläche erstellt werden die es erlaubt, sich mit einem Nickname an einem Server anzumelden und Nachrichten zu verschicken.

Aufgabe 1: Projekt erstellen (2P)

Erstellen Sie ein Eclipse Projekt, welches die folgenden vier Java Packages im `src` Verzeichnis enthält:

- de.uni_kassel.pmss2010.ha11.client
- de.uni_kassel.pmss2010.ha11.client.controller
- de.uni_kassel.pmss2010.ha11.client.gui
- de.uni_kassel.pmss2010.ha11.server

Aufgabe 2: Server erstellen (10P)

Erstellen Sie eine Klasse `Server` im Package `de.uni_kassel.pmss2010.ha11.server`. Der Server hat die Aufgabe an einem bestimmten Port auf eingehende Verbindungen zu hören. Der Port soll dem Server als Parameter beim Starten übergeben werden können.

Erstellen Sie danach eine Klasse `ConnectionHandler` im gleichen Package, die von der Klasse `java.lang.Thread` erbt. Für jede beim Server eingehende Verbindung wird ein neuer `ConnectionHandler` erzeugt und gestartet, der fortan Nachrichten des Clients entgegennimmt und dem Server übergibt. Der Server schickt dann die Nachricht an alle anderen Clients.

Wird die vom `ConnectionHandler` gehaltene Verbindung unterbrochen, soll der Server von nun an keine Nachrichten mehr an diesen Client verschicken.

Hinweis: Sie können sich bei der Implementierung an der PM Übung 13 orientieren.

Aufgabe 3: Client erstellen (5P)

Erstellen Sie eine Klasse `ClientNetworkHandler` im Package `de.uni_kassel.pmss2010.ha11.client` welche von der Klasse `java.lang.Thread`

erbt. Diese soll im Konstruktor den Host (als `String`) sowie den Port übergeben bekommen. Wird der Handler gestartet, nimmt er Verbindung mit dem Server auf. Von nun an empfängt er vom Server kommende Nachrichten und gibt sie über `System.out.println()` aus.

Erstellen Sie eine Methode `sendMessage(message:String):void`, die eine als Parameter übergebene Nachricht an den Server schicken kann.

Aufgabe 4: Login GUI (5P)

Erstellen Sie im Package `de.uni_kassel.pmss2010.ha11.client.gui` eine grafische Oberfläche mit der man sich an einem Server anmelden kann. Orientieren Sie sich dabei an dem in Abbildung 1 dargestellten Mockup.

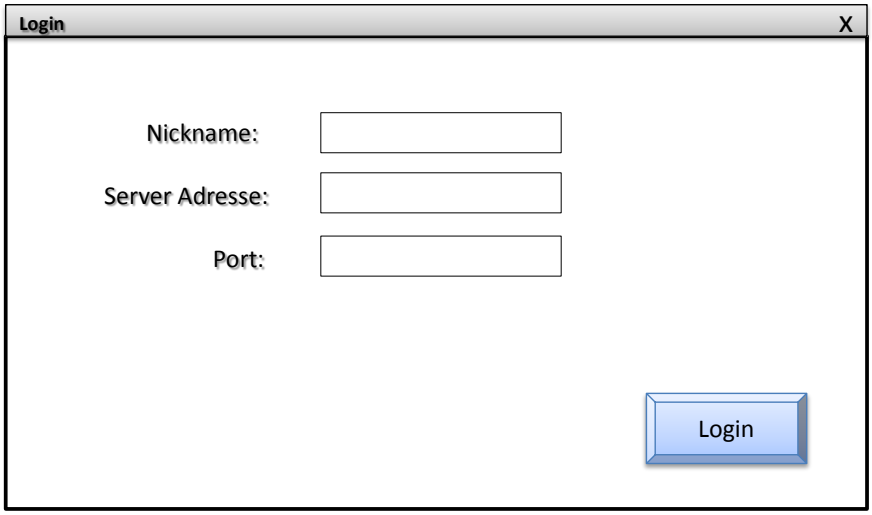

Abbildung 1: Mockup zum Login

Aufgabe 5: Chat GUI (5P)

Erstellen Sie im Package `de.uni_kassel.pmss2010.ha11.client.gui` eine grafische Oberfläche mit der man Nachrichten an einen Server verschicken und empfangen kann. Orientieren Sie sich dabei an dem in Abbildung 2 dargestellten Mockup.

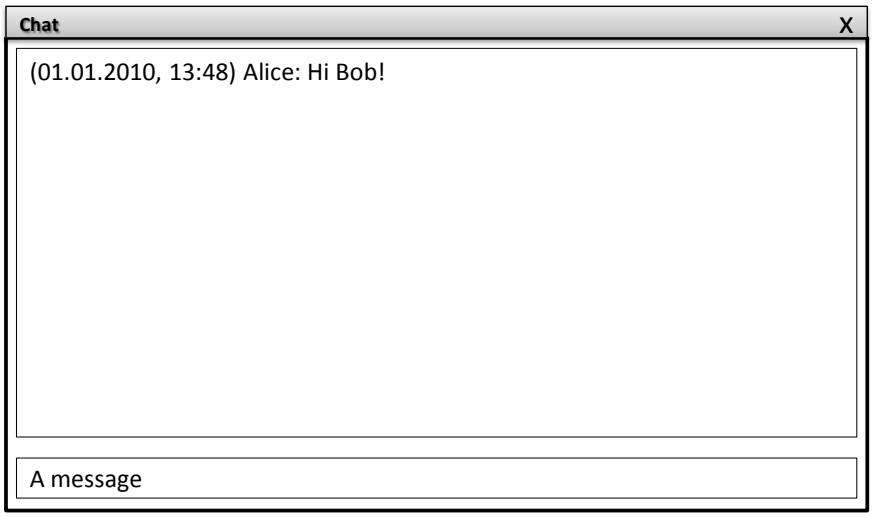

Abbildung 2: Mockup zum Chat

Aufgabe 6: LoginController (5P)

Erstellen Sie im Package `de.uni_kassel.pmss2010.ha11.client.controller` einen Controller, der zunächst sinnvolle Startwerte für die Felder „Server Adresse“ (z. B. „localhost“) und „Port“ (z. B. „5000“) festlegt. Klickt der Benutzer auf „Login“, soll zunächst eine Instanz der in Aufgabe 5 erstellten Chat GUI erzeugt und dann an den `ClientController` übergeben werden. Dann kann der `LoginController` gestoppt und der `ClientController` gestartet werden.

Aufgabe 7: ClientController (5P)

Erstellen Sie im Package `de.uni_kassel.pmss2010.ha11.client.controller` einen Controller, der im Konstruktor die in Aufgabe 5 erstellte Chat GUI, sowie die in die Login GUI eingegebenen Werte für „Nickname“, „Server Adresse“ und „Port“ übergeben bekommt.

Der `ClientController` ist dafür zuständig die Verbindung zum Server über den in Aufgabe 3 erstellten `ClientNetworkHandler` aufzubauen. Empfängt der `ClientNetworkHandler` eine Nachricht vom Server, soll diese in den oberen Teil der Chat GUI (`JTextField`) eingefügt werden. Der Benutzer kann einen Text in den unteren Teil der Chat GUI (`JTextField`) eingeben und mit `ENTER` abschicken. Hierzu delegiert der `ClientController` diese Aufgabe an den `ClientNetworkHandler`.