APR Designüberlegungen

Es soll ein Auktions-System mit Server und Clients erstellt werden. Der Server verwaltet die Auktionen und die Clients bieten auf die Transaktionen.

Folgende Aufgaben haben die entsprechenden Teile:

## Client

* Login am Server mittels !login <name>, zu realisieren mittels einer login(name) Methode, wobei die Syntax am Client überprüft wird und nur bei Korrekter gesendet wird. Variable loggedIn -> true
* Logout mittels !logout-> Lokales Prüfen ob eingeloggt, wenn ja senden des Befehls an Server, Warten auf Antwort, **Beenden** des Threads der auf UDP-Pakete wartet, **Beenden** Thread der die TCP-Connection zum Server instand hält.
* !list -> Auflisten aller Auktionen, läuft über die TCP-Verbindung, ohne anmelden möglich
* !bid <id> <amount> -> Bietet eine gewisse Anzahl auf eine Auktion; Bei Erfolg bekommt der Client eine gewisse Rückmeldung, bei Misserfolg ebenso

**Netzwerk:**

Der Client muss folgende Threads und Sockets haben, um mit dem Server kommunizieren zu können:

* Start-Thread, dieser baut eine TCP-Verbindung mit dem Server auf. Er behandelt auch die Eingaben des Users und sendet sie an den Server weiter. Während der Kommunikation kann der User nichts eingeben, da geblockt wird, dies ist ein gewünschter (!) Effekt
* Thread der auf UPD-Pakete des Servers wartet. Wird eins empfangen, werden die Daten weitergeleitet. **Problem:** Ausgabe könnte von anderem Thread besetzt sein

Beim Starten des Clients müssen folgende Argumente angegeben werden:

* host: host name or IP of the auction server
* tcpPort: TCP connection port on which the auction server is listening for incoming connections
* udpPort: this port will be used for instantiating a java.net.DatagramSocket (handling UDP notifications from the auction server).

# Server

Der Server empfängt hauptsächlich Pakete von Clients und bearbeitet diese. Er benötigt folgende Komponenten:

* Liste aller User (mit IP, Port)
* Liste aller Transaktionen (wird als Objekt realisiert)
* Thread der auf eingehende Verbindungen eingeht (Realisiert Anmelden von Clients und leitet sie weiter)
* Pro Client ein Thread (aus Thread-Pool) – TCP, nimmt Anfragen des Clients war. Greift auf die Transaktionen zu -> Read/Write Lock notwendig
* Ein Thread für UDP-Verbindungen. Dieser kann UDP-Pakete seriell schicken und muss nicht durch mehrere Threads realisiert werden
* Thread pro Transaktion (?) / UDP-Thread fragt Ta ab?

# Mögliche Probleme

|  |  |
| --- | --- |
| Problem | Lösung |
| Auflisten von TA als nicht angemeldeter User, welcher Thread wird verwendet |  |
| Nebenläufiges Schreiben von Transaktionen | Read/Write Locks |
| Client: Die Ausgabe könnte von einem anderen Thread besetzt sein |  |
| Versenden von Nachrichten an nicht angemeldete User | Ev. Im Thread warten bis User erneut angemeldet ist. Nachrichten in einer Liste vom User schreiben |
| User beendet ohne Logout | TCP-Timeout? Variable beim User ob aktiv, wenn nicht Nachrichten in Liste |
|  |  |
|  |  |

# Verwendete Klassen

* Client
  + Main-Thread
  + PaketReceiver
* Server
  + ConnectionReceiver
  + Broadcaster
  + ClientHandler (in Liste)
  + AuctionHandler
* Auktion mit: ID, Name, Beschreibung, Datum, bidsize=0, Höchstbieter
* User mit: Port, IP, Name
* Testklassen