|  |
| --- |
|  |

|  |
| --- |
| **Task 08** |
| RMI Auctionsystem |
|  |
|  |
|  |
| **Krepela, Lipovits, Reichmann, Tattyrek, Traxler** |
| **29.01.2014** |
|  |

Insert Specification Here

Designüberlegung

# Testing Component

**Reading Property File:**

# TODO: adjust these values

clients = 100

auctionsPerMin = 1

auctionDuration = 2\*60

updateIntervalSec: 20

bidsPerMin = 2

Lines, which start with a '#' are comments and do not affect any functionality.

The next line describes the number of clients which should be used within this test.

Afterwards, the auctions per minute, the auction duration, the update interval in seconds and the bids per minute are given.

The attribute and the value can be split by '=' or ':', additionally the number can consist of two multiplicators, which have to be multiplicated before it can be saved.

**Exceptions (Lipovits only)**

CommandNotFoundException()

→ Thrown if a Command does not exist

IllegalNumberOfArgumentsException()

→ Thrown, if the userinput conists of a wrong number of arguments for the command

WrongInputEception()

→ Thrown, if the command exists and has the right number of arguments, but one or more arguments are of a wrong type. e.g. '!removeSteps 1 miau'

# Management Client

**Commands**

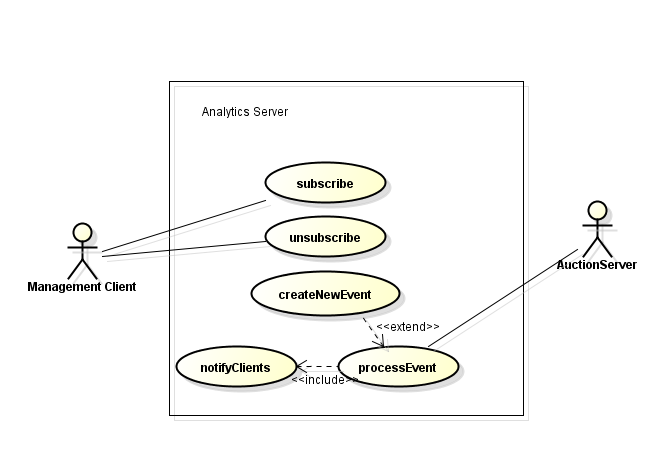
The following management client - commands where implemented as a prototype:

* !login
* !logout
* !steps
* !addStep
* !removeStep
* !bill
* !subscribe
* !unsubscribe

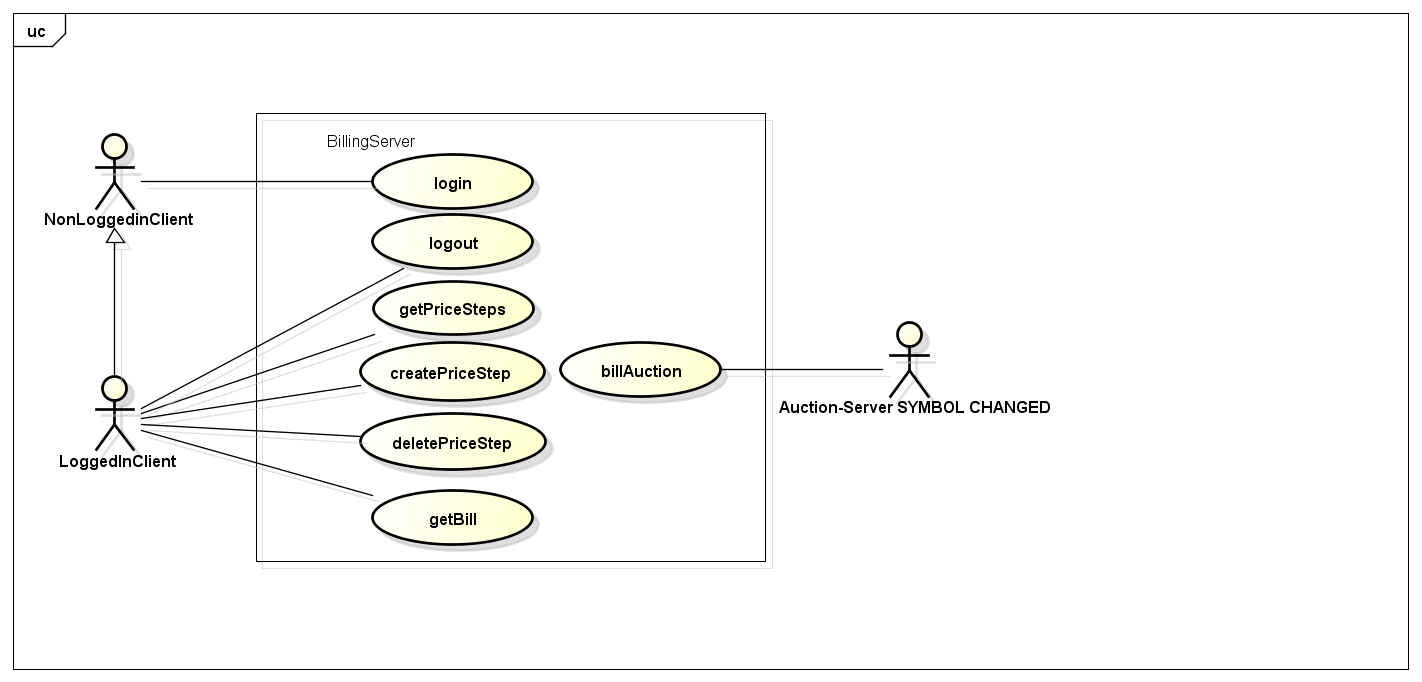
Those commands are recognized and checked by the client and print a response.

## Use Case

Analytic Server:



BillingServer:



## AktivitätsdiagrammAuktionen anzeigen.jpgAuktionsende.jpgBieten.jpgClient Login.jpgClient Logout.jpgEinschreiben.jpgErstellen .jpgEvent verarbeiten.jpgManagement Login.jpgManagement Logout.jpgStep bearbeiten.jpgSteps anzeigen.jpg

## Klassendiagramme

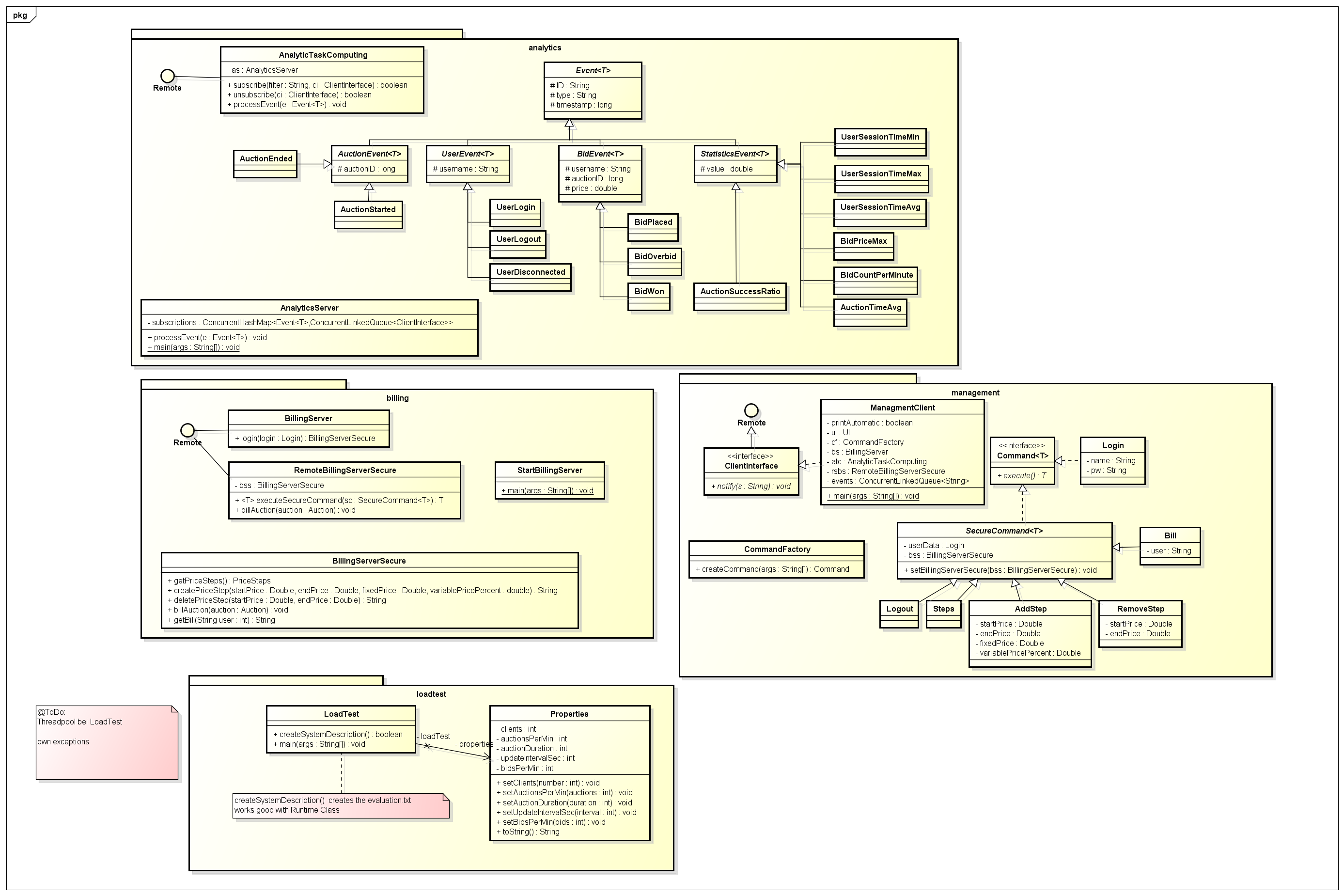
### Analytic + Events

### C:\Users\Daniel\Desktop\Class Diagram2.pngBilling

### C:\Users\Daniel\Desktop\Class Diagram2.png

### ManagemenclientC:\Users\Daniel\Desktop\Class Diagram2.png

### Testing Component



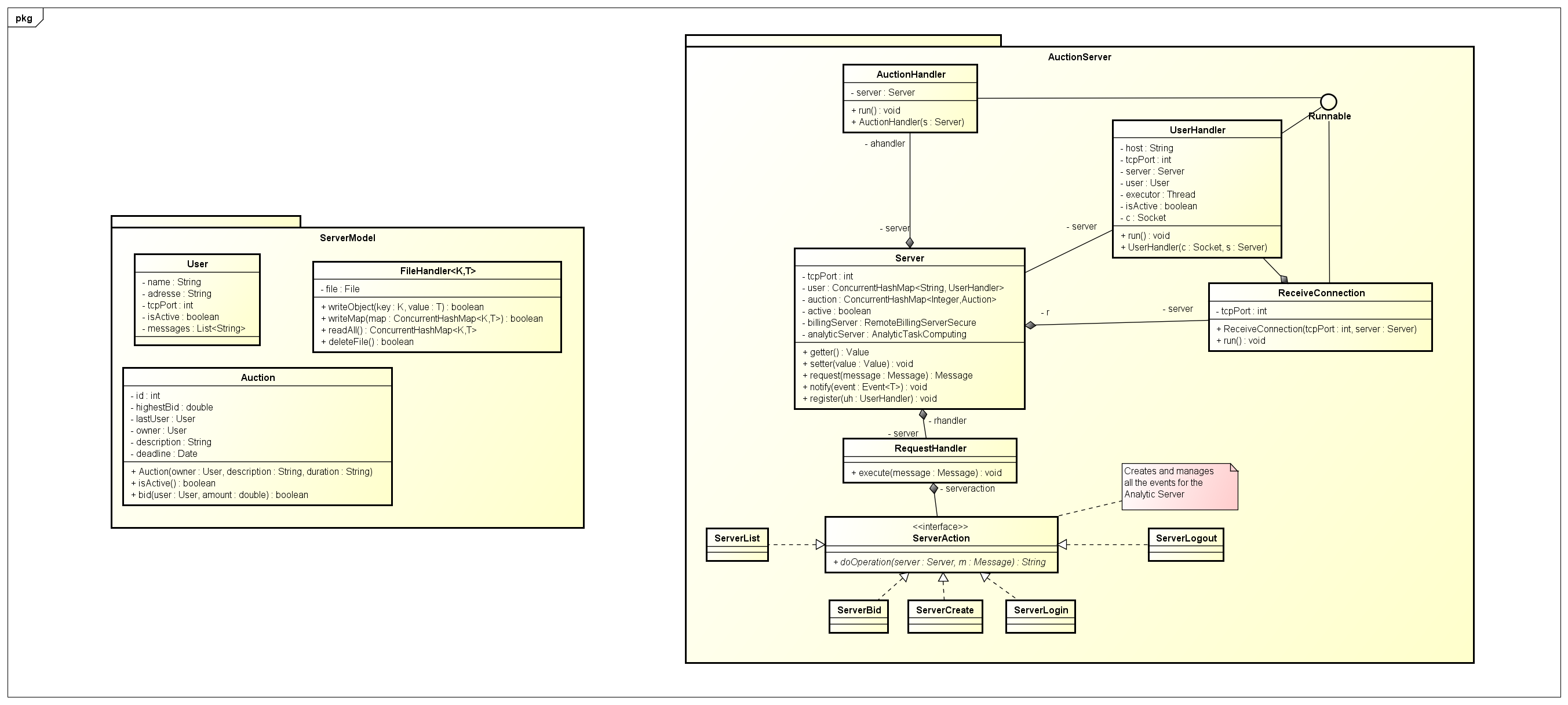
### Client



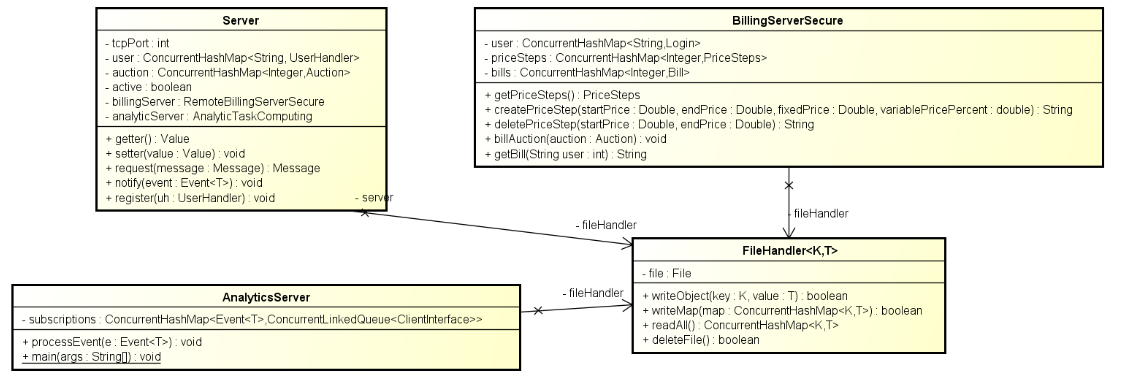
### Message-Model



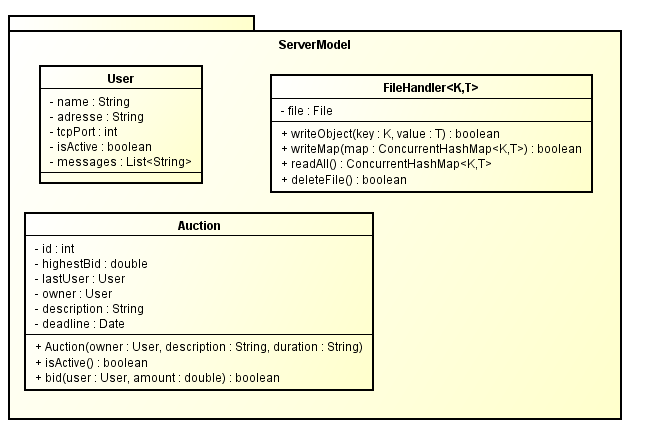
### Server



### FileHandler



### Model Server



# Zeitschätzungen und Arbeitsaufteilung



# Absprachen

-> Analytic Server

-> Billing Server (Frage GRAFIK?) schritte setzen, abrechnung erstellen -> keine persistenz

-> Testing Load Client (Viele Clients machen bids etc.)

-> Management Client (Befehle für Billing Server, Benachrichtungungen etc)

Altes Programm → List-> ConcurrentHashMap, Eigene Exceptions → UDP Notification brauchen wir nicht mehr

UML-Klassendiagramm → Krepela, Traxler

Aktivitätdiagrann → Lipovits

Use Case Diagramme → Reichmann

Checker → Tattyrek

Tasks:

* RMI-Verbindungen → Traxler
* Analytics Server → Reichmann, Tattyrek
* Billing Server → Krepela
* Management Client → Lipovits
* Testing Component → Lipovits
* Model →Tattyrek (Model JUnitTests)
* Ausbesssern alten Code → Traxler, Reichmann
* Ant,Protokoll → Reichmann

JEDER TESTET SEINEN TEIL + TECHNOLOGIEBESCHREIBUNG FÜRS PROTOKOLL!!!