|  |
| --- |
|  |

|  |
| --- |
| **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**

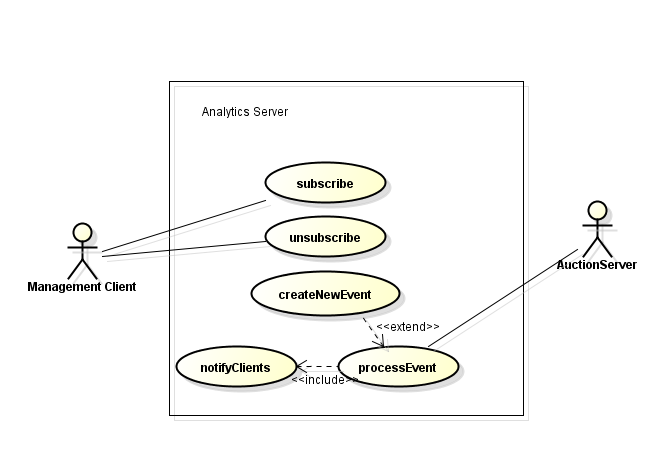
The following management client - commands where implemented as a prototype:

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

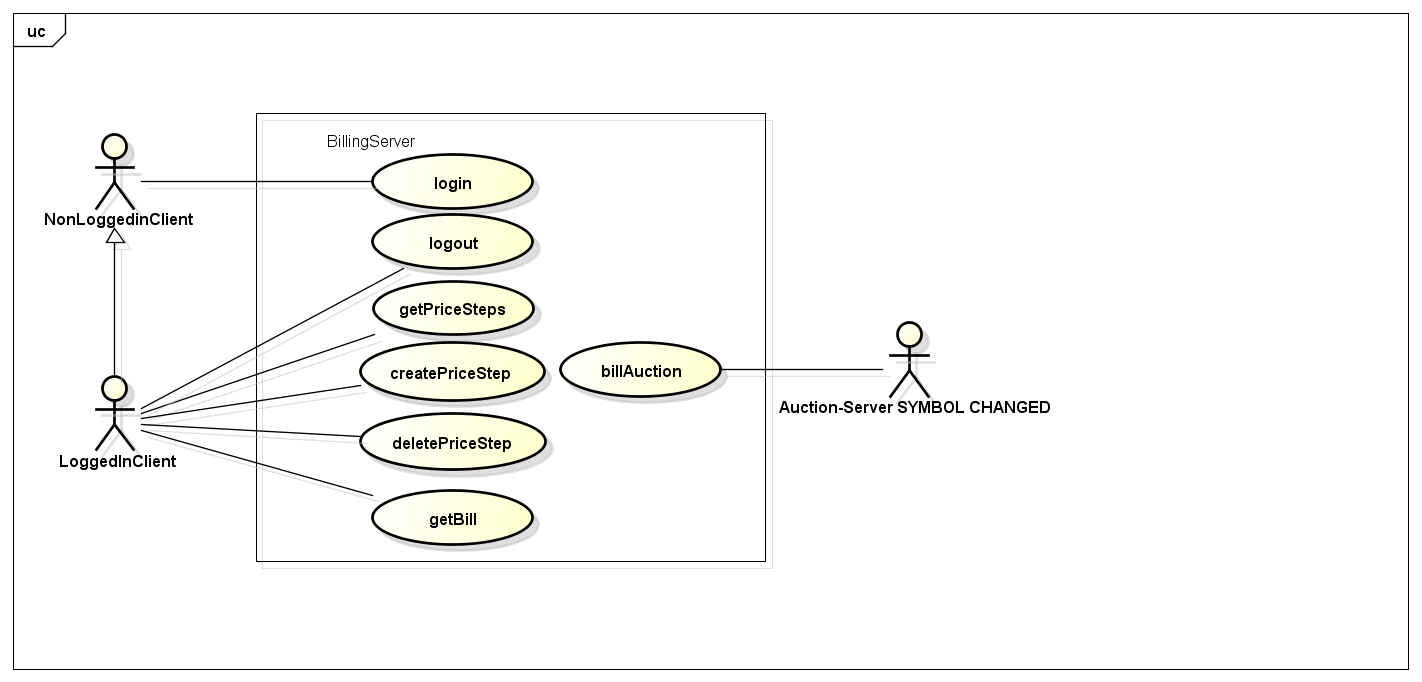
Those commands are recognized and checked by the client and print a response.

## 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

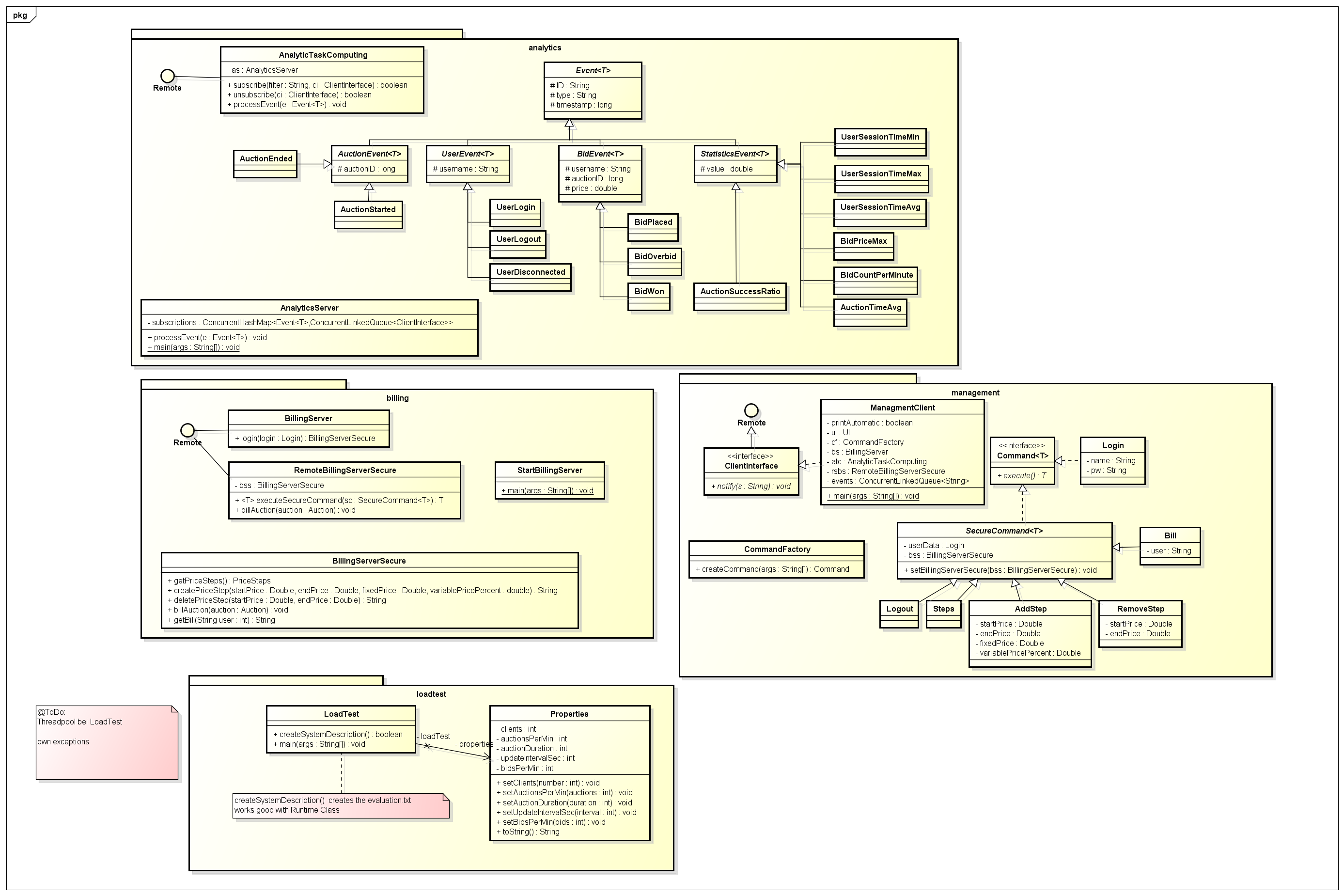
### Analytic + Events

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

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

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

### Testing Component



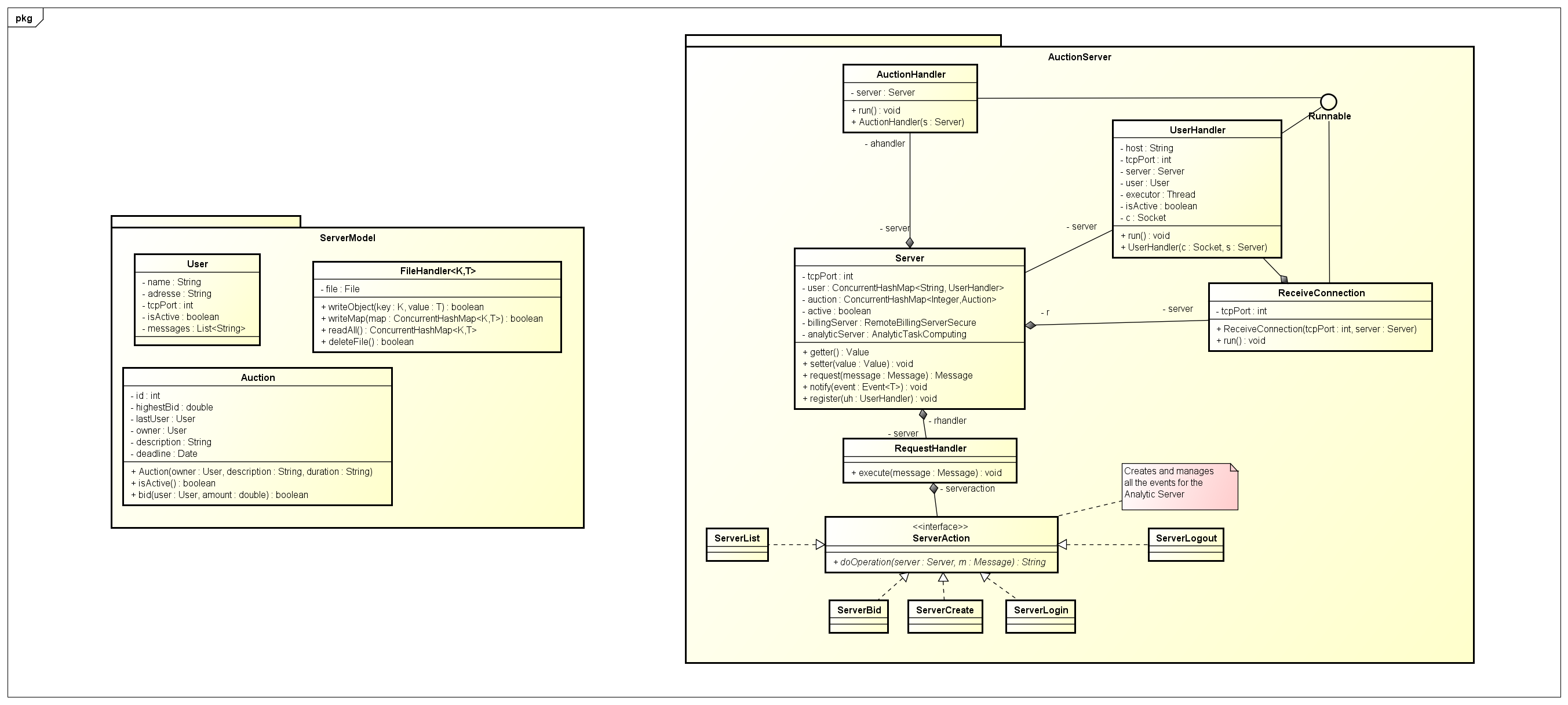
### Client



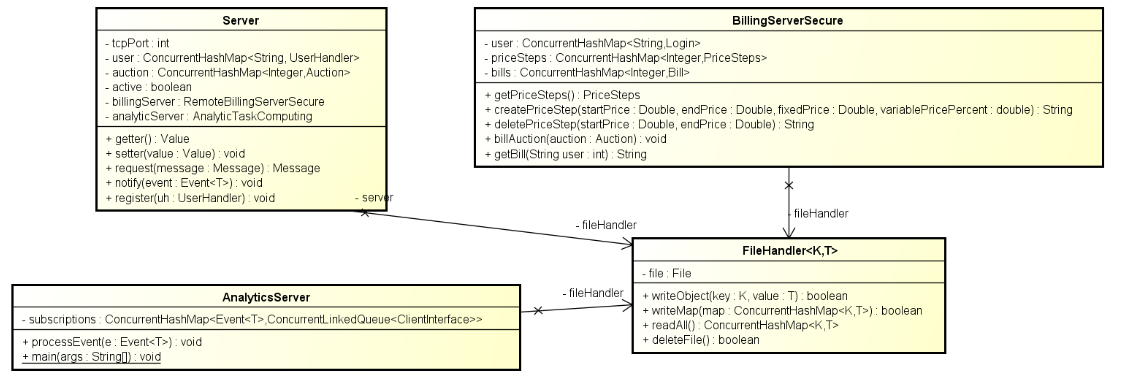
### Message-Model



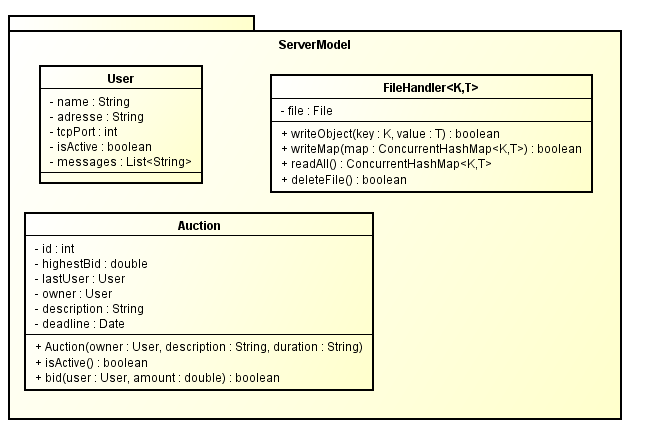
### Server



### 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!!!

**Arbeitspakete bis Montag, 10. Februar**

Liebes Team,

damit auch was beim Projekt weitergeht, wird es notwendig die Aufgaben zu definieren:

Jeder sollte bis zum 10. Februar einen Prototypen (= 70% Funktionalität) umgesetzt haben. Dieser soll bis Montag fertig sein, da ich einen Tag benötige die Arbeitspakete zu kontrollieren und etwaige Sachen auszubessern. Ich befinde mich selbst auf Urlaub, werde jedoch per Mail erreichbar sein und kann auf auftretende Fragen antworten.

Die betreffenden Arbeitspakete pro Person könnt ihr aus der Liste auslesen. Bitte haltet euch bei der Umsetzung an das UML und schreibt mir wenn(am besten bevor) ihr Änderungen an diesem machen müsst.

Bitte sprecht euch mit eurem jeweiligen Mitarbeiter ab, wer welchen Teil übernimmt. Es gibt pro Server einen Hauptverantwortlichen der im Notfall das Sagen hat.

Dokumentiert bitte eure Vorgehensweisen, damit wir später ein gutes Protokoll haben.

**Wichtig:** Da wir vom Borko die Tests durchgeführt bekommen, muss die Funktionalität bis zum 17. Gegeben sein! Sonst haben wir nichts von unserem Vorteil. Unit-Tests bitte auch beachten, können aber bis zum 20. Fertiggestellt werden. Und: Ihr dürft ruhig mehr machen, dann sind wir halt schneller fertig^^

**Gewünschter Fortschritt bis zur Deadline:**

Management-Client: (Lipovits)

* Usereingaben werden erkannt (alle geforderten Befehle) und auf Syntax kontrolliert (Trockene Ausgabe ohne Senden)
* Bei falschen Eingaben wird eine **eigene** Exception geworden (außer nicht beötigt)

Load-Testing Component: (Lipovits)

* Properties aus dem File lesen und entsprechend speichern
* Liste an Clients (Package Client aus altem src) erstellen
* Implementieren Fake\_Cli (Stichwort InputStream)

BillingServer + Secure (Krepela)

* Erstellen von PriceSteps
* Löschen von PriceSteps
* Anzeigen der PriceSteps schön formatiert
* Anmelden mittels Username + pwd, auslesen aus Property-File (siehe Angabe); Vermittlung muss noch nicht erfolgen
* Rechnung erstellen lassen

AnalyticsServer (Reichmann)

* Alle Events als Models erstellt und entsprechende HashMap angelegt
* Dokumentieren der Abhängigkeiten (bei welchem Eingehenden Event werden welche erzeugt)
* Berechnung von neuen Events
* Passende Events zu einem Regex finden
* Überlegen von Notifications zu einem Event

RMI-Verbindungen (Traxler)

* Alle Interfaces definiert
* Stubs implementiert
* Sozusagen fertig um die Componenten nurnoch verbinden zu müssen

Alter Server: (Traxler/Reichmann)

* ArrayList durch HashMap ersetzen
* Eigene Exceptions definieren
* ThreadPool einsetzen
* Ev TimerService

**Diagramme:** (Reichmann)

* Kein Schatten
* Include bei Use-Case entfernen
* Sichtbarer machen

**Mfg Daniel**

P.S.: Wenn euch die Aufgaben zu viel sind, dann bitte ich um Rückmeldung, damit ich das eine oder andere Arbeitspaket überdenken kann.