

An abstract graphic featuring three blue circles of varying sizes and three thin blue lines. One line connects the top-left of the largest circle to the top-left of the medium circle. Another line connects the top-right of the largest circle to the top-right of the medium circle. A third line extends from the bottom-right of the largest circle towards the bottom-right of the page.

# 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 multipliers, which have to be multiplied before it can be saved.

### Exceptions (Lipovits only)

CommandNotFoundException()

→ Thrown if a Command does not exist

IllegalNumberOfArgumentsException()

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

WrongInputException()

→ 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 were 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.

## Billing Server

### Saving price steps:

`ConcurrentHashMap<CompositeKey,PriceStep> priceSteps;`

CompositeKey has 2 attributes (pk's): start and end price. It also provides methods to compare these keys. The function `overlaps()` tests if they collide with each other.

Price Step contains all 4 attributes.

### Exceptions

`IllegalArgumentException ()`

→One or more arguments are invalid!

e.g. :

values below zero

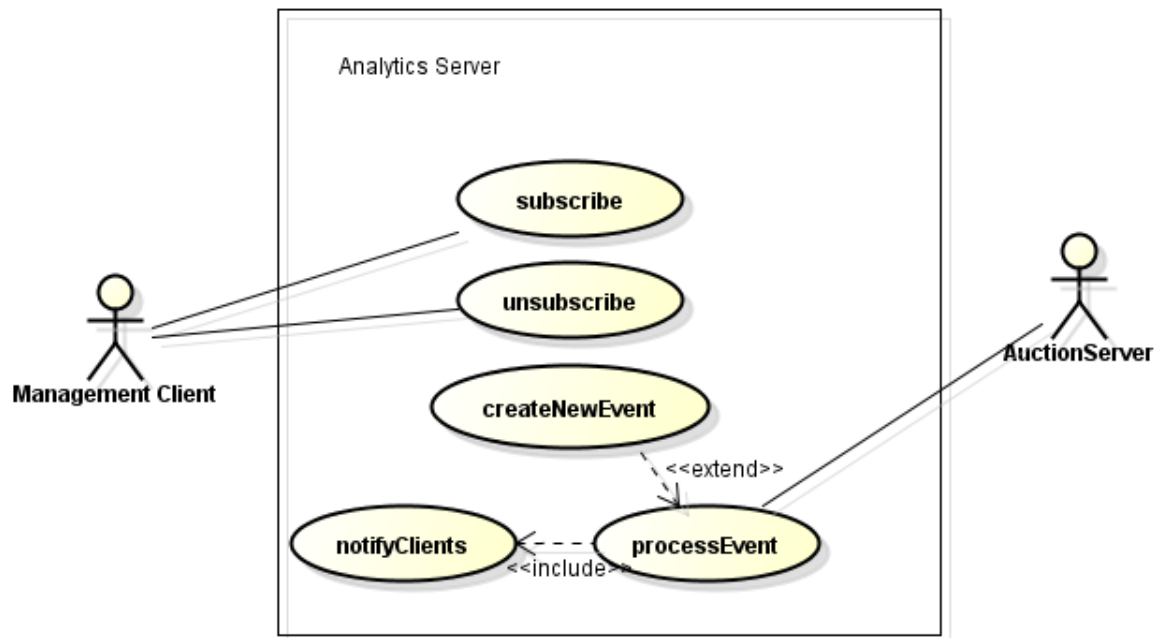
endprice must be bigger than startprice

`PriceStepIntervalOverlapException()`

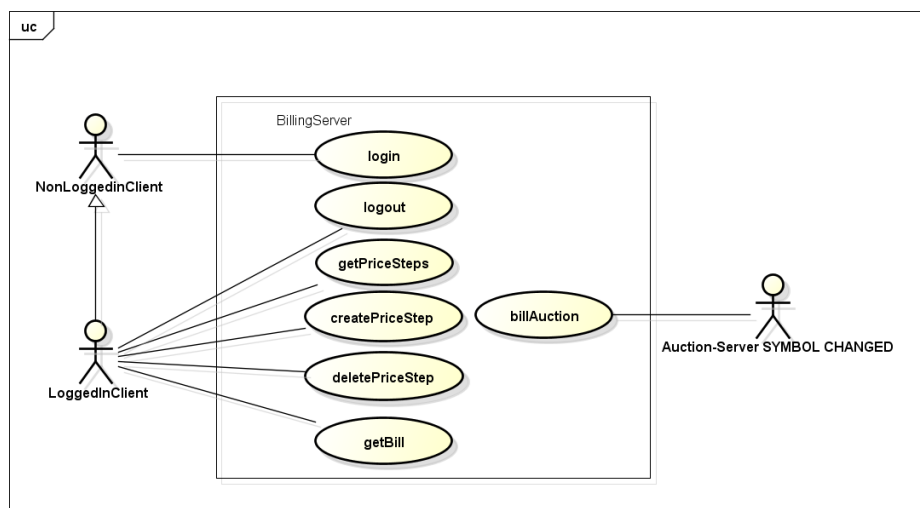
→The provided price interval overlaps with an existing price step  
(delete the other price step first)!

## Use Case

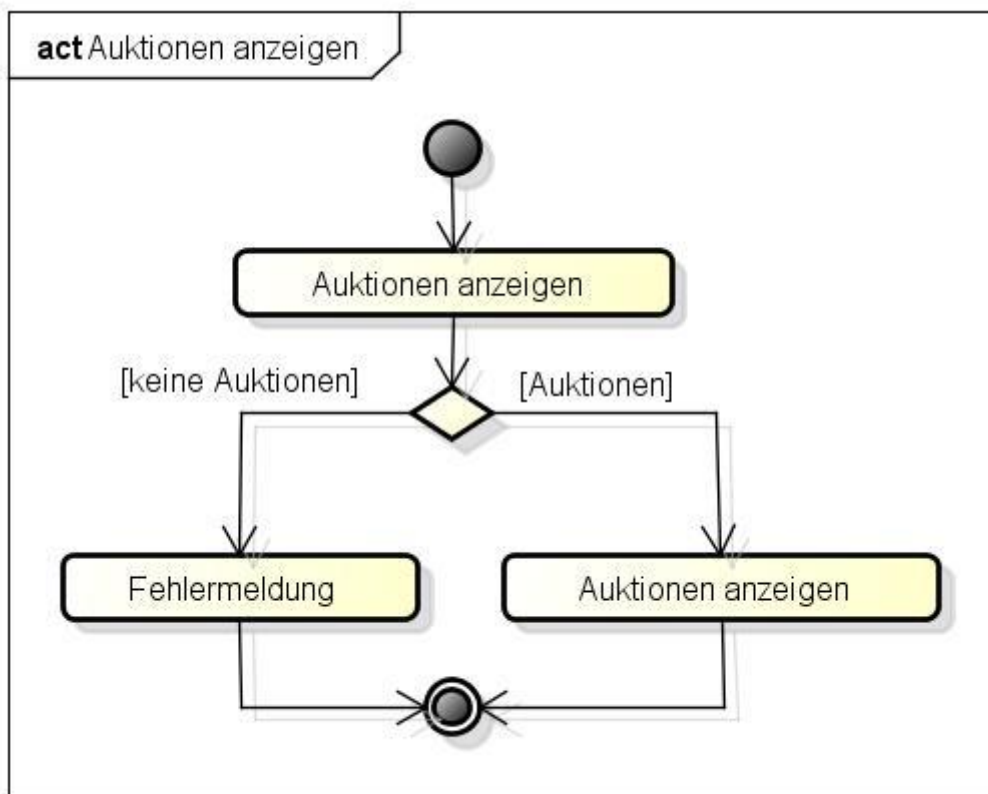
Analytic Server:



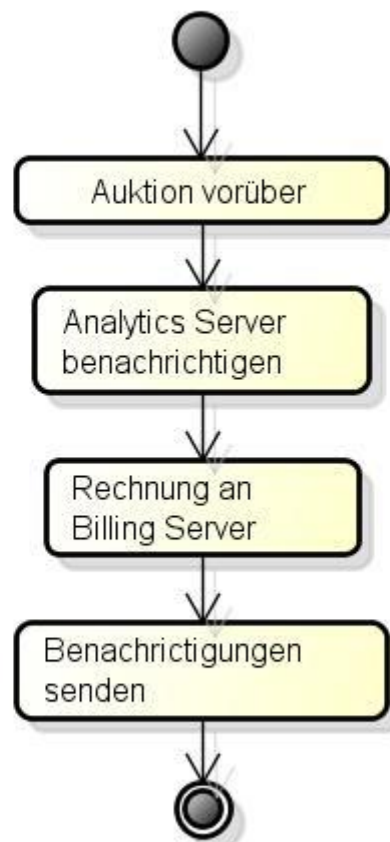
BillingServer:



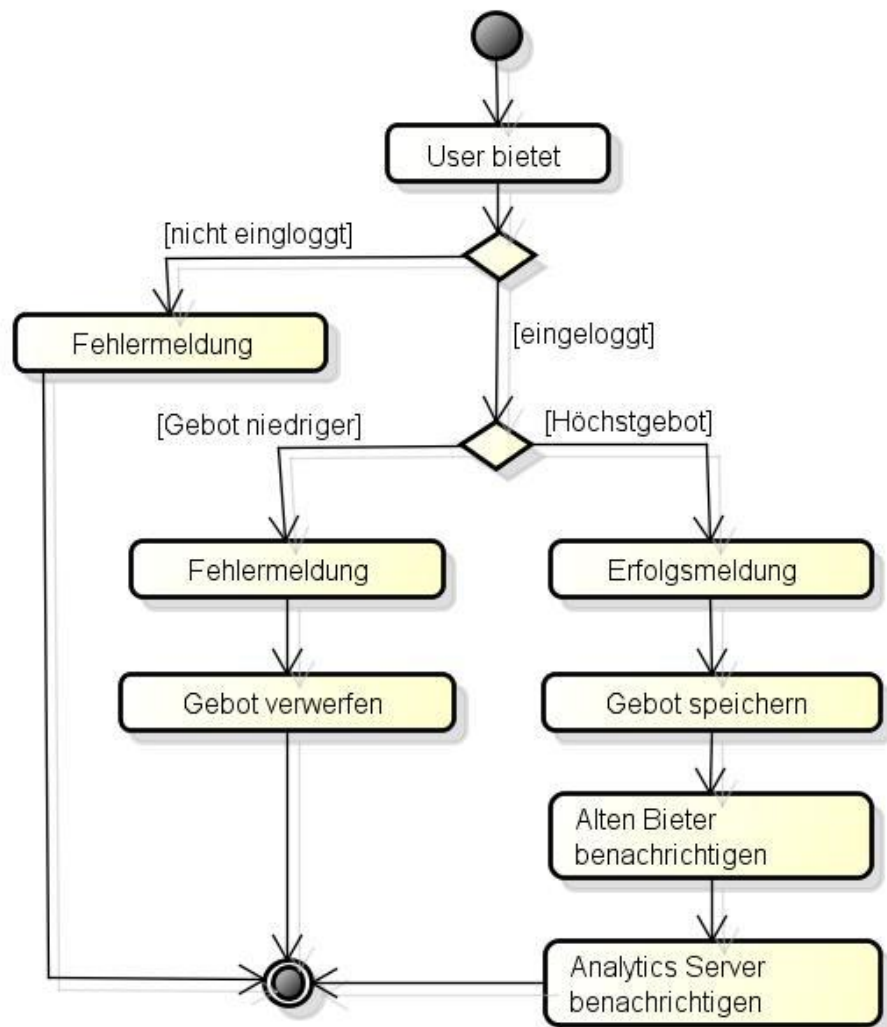
## Aktivitätsdiagramm



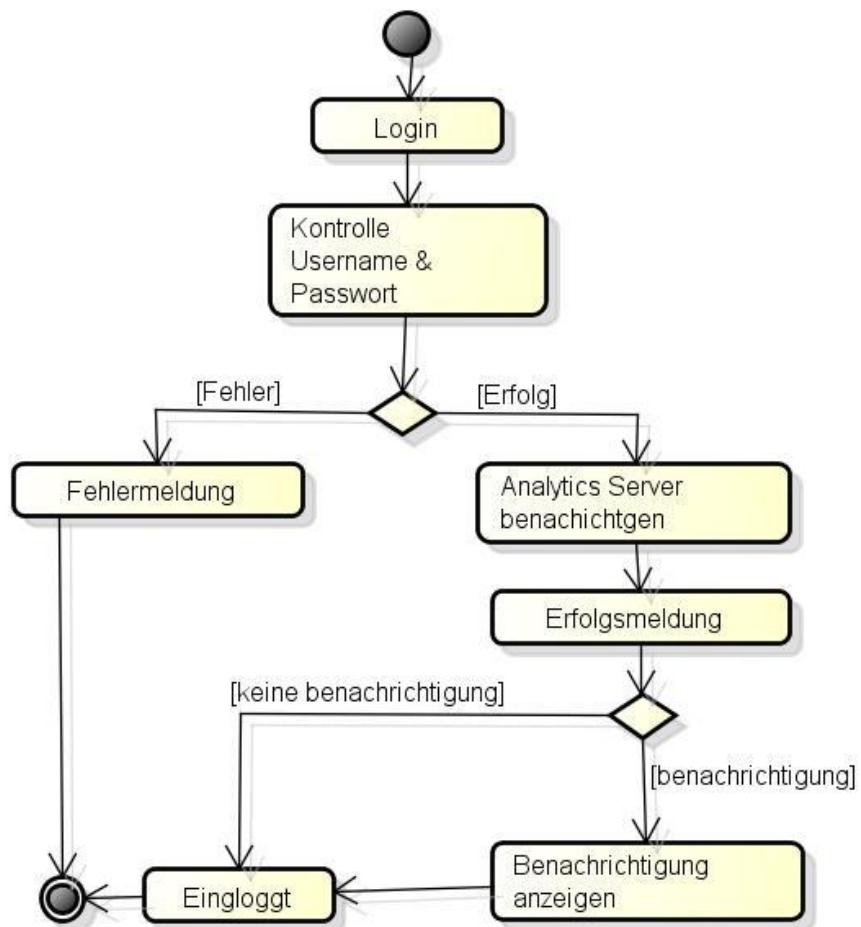
**act** Auktionsende



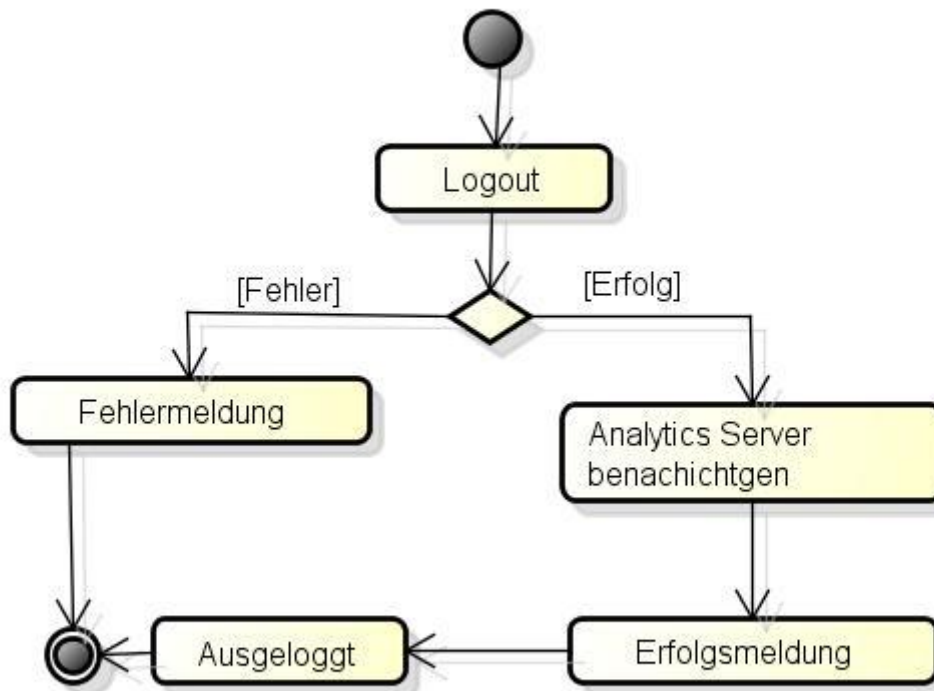
**act**Bieten



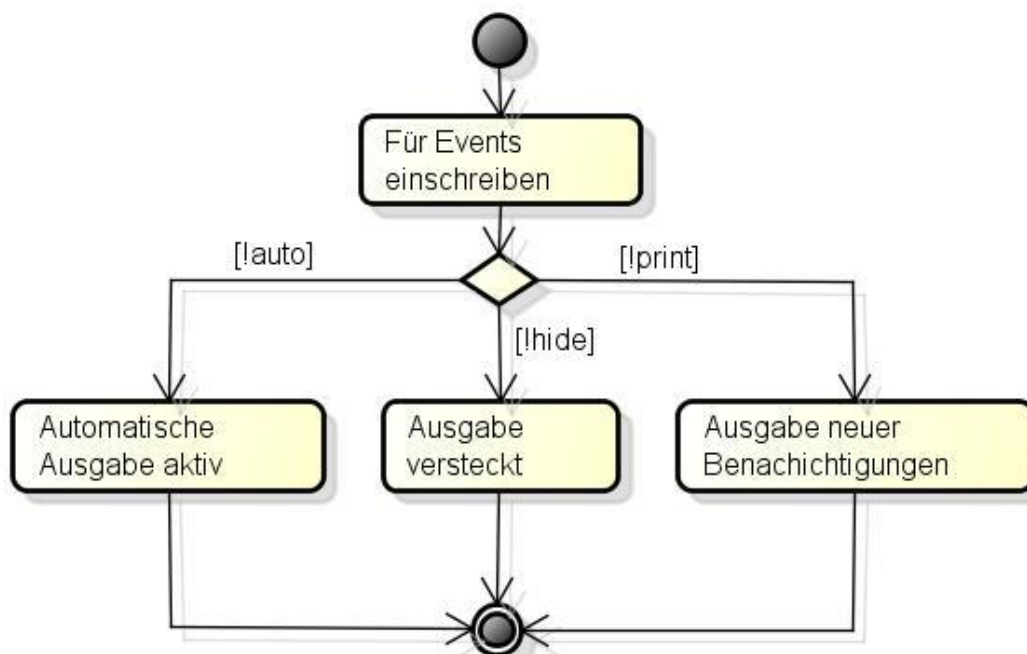




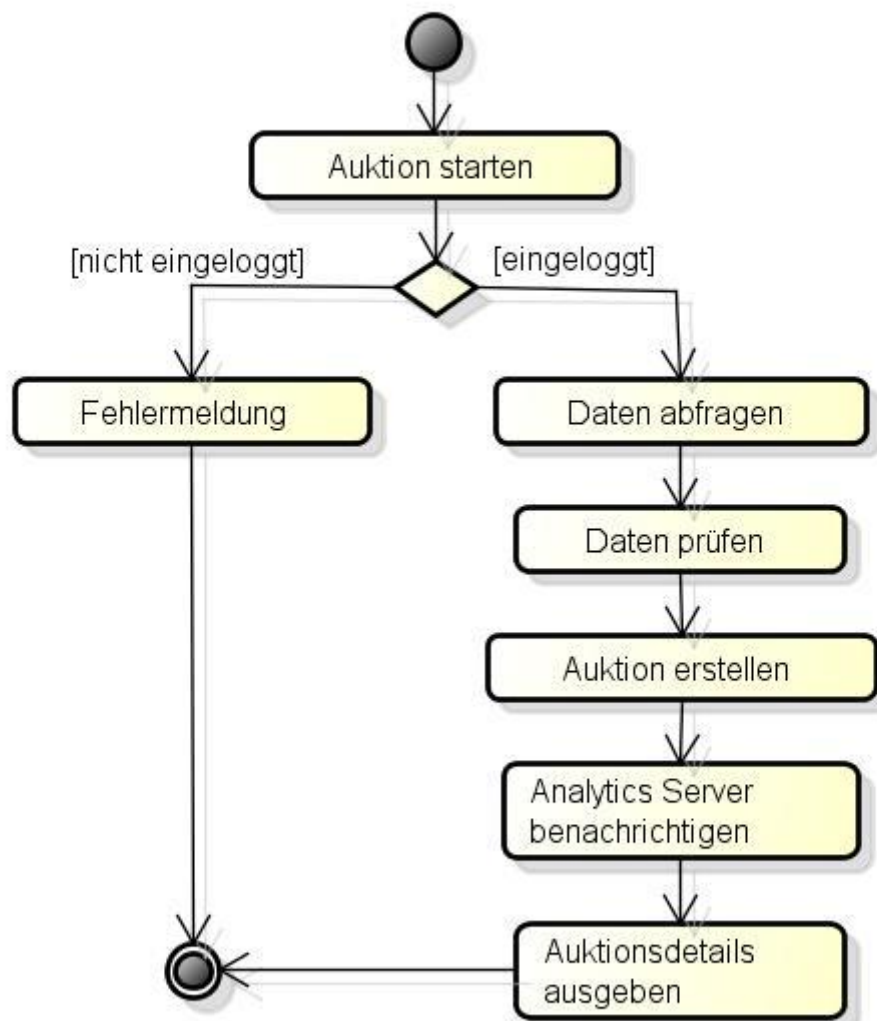
### act Client Logout



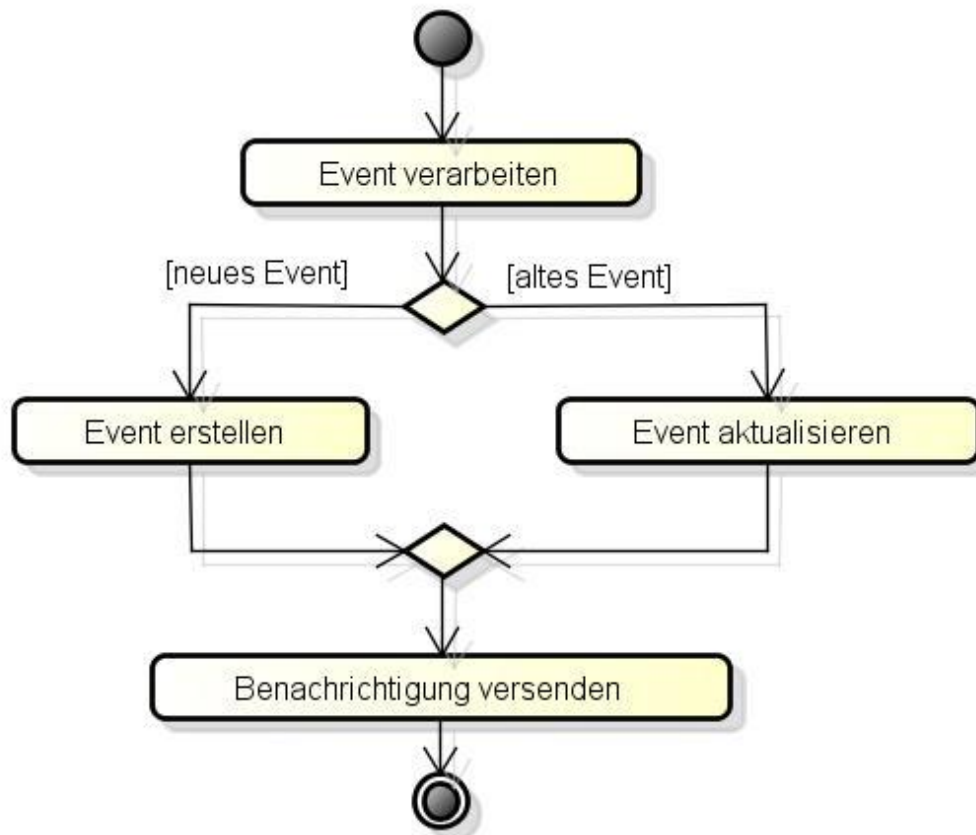
### act Einschreiben



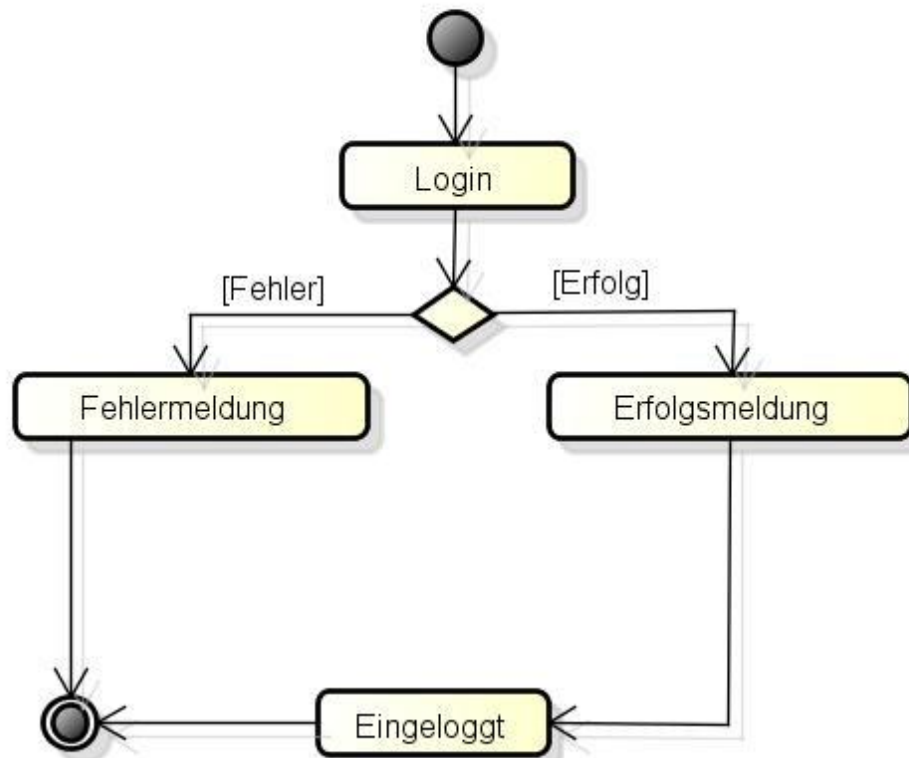
## actErstellen



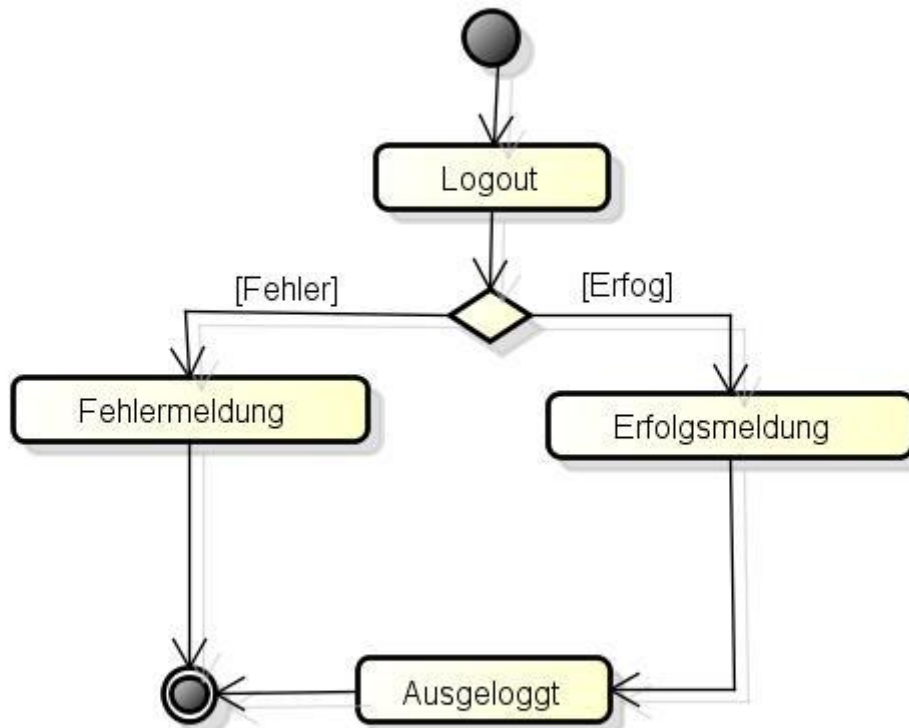
**act**Event verarbeiten



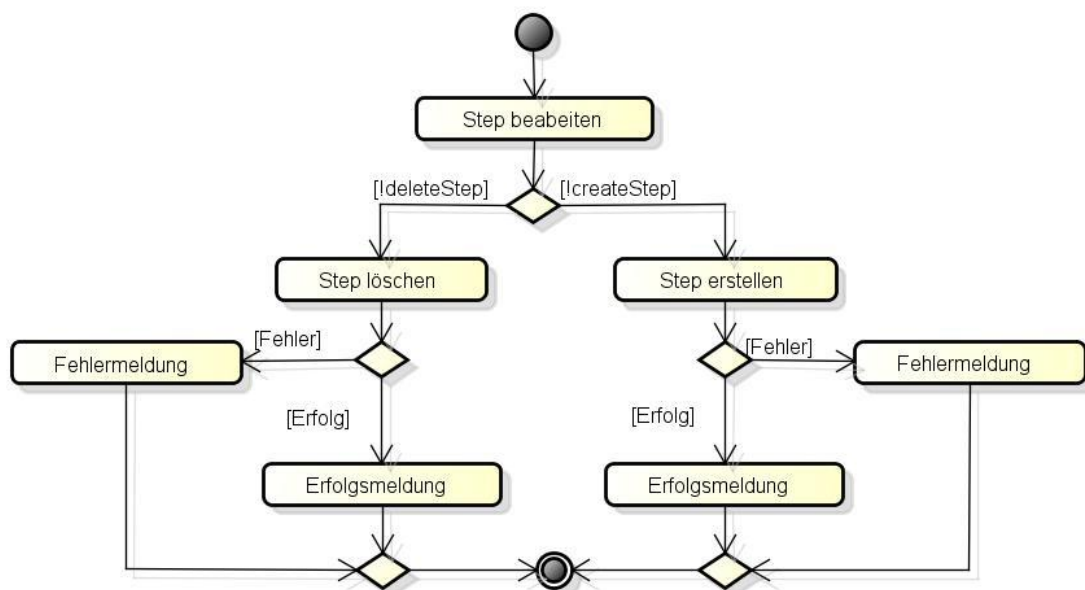
**act**Management Login



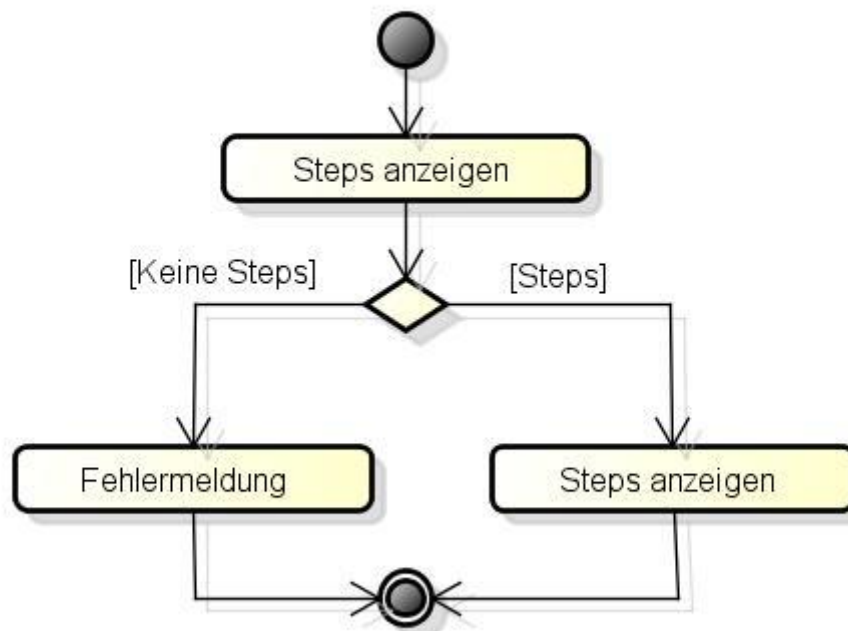
### actManagement Logout



### actStep bearbeiten

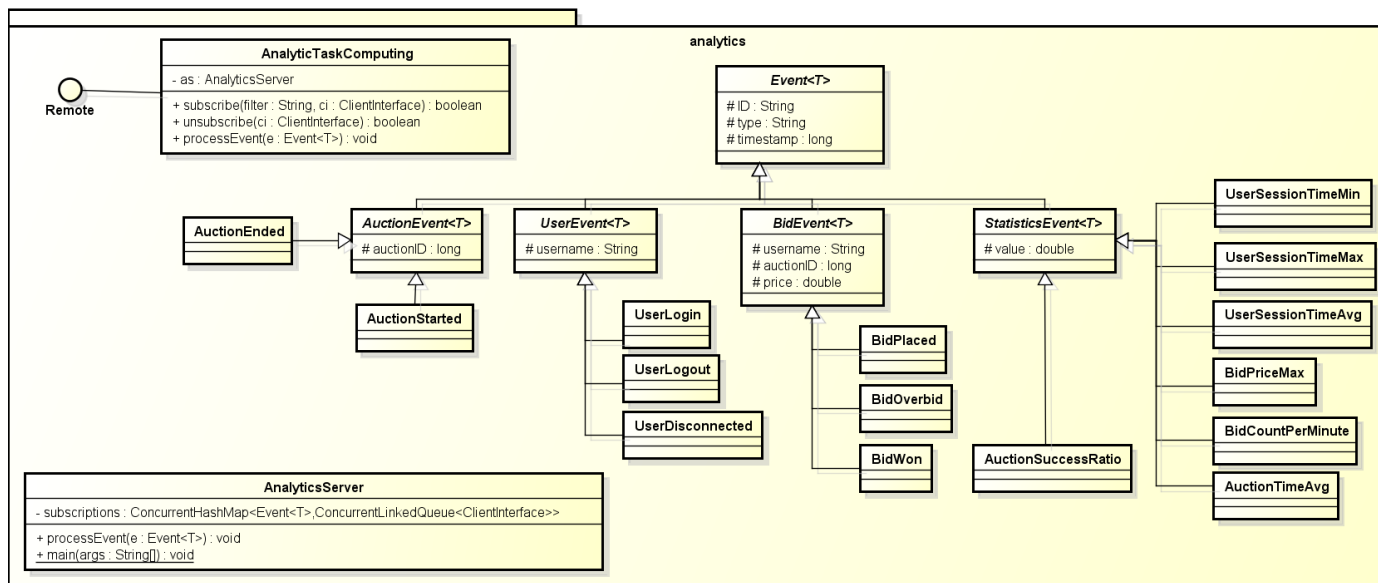


**act** Steps anzeigen



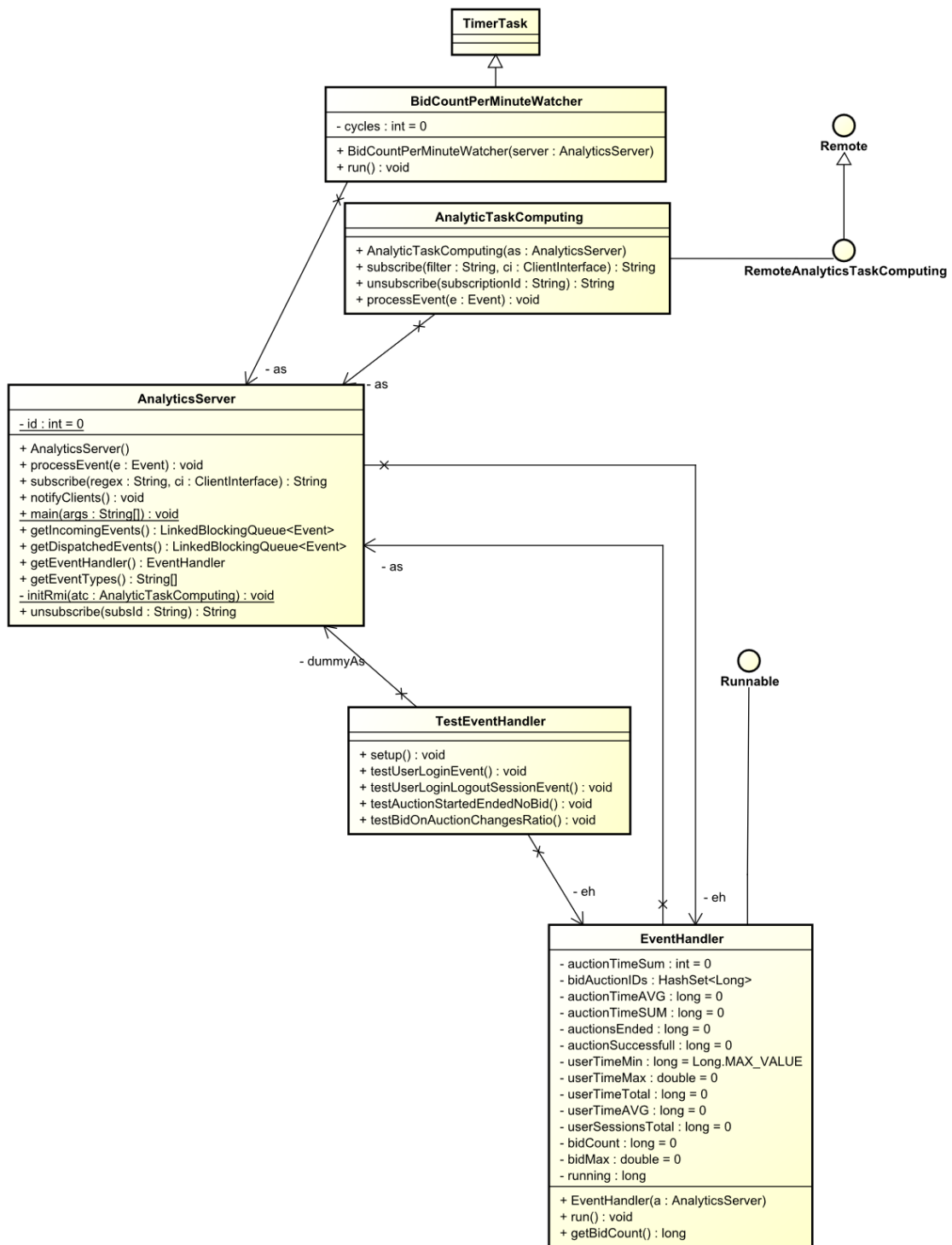
# Klassendiagramme

## Analytic + Events

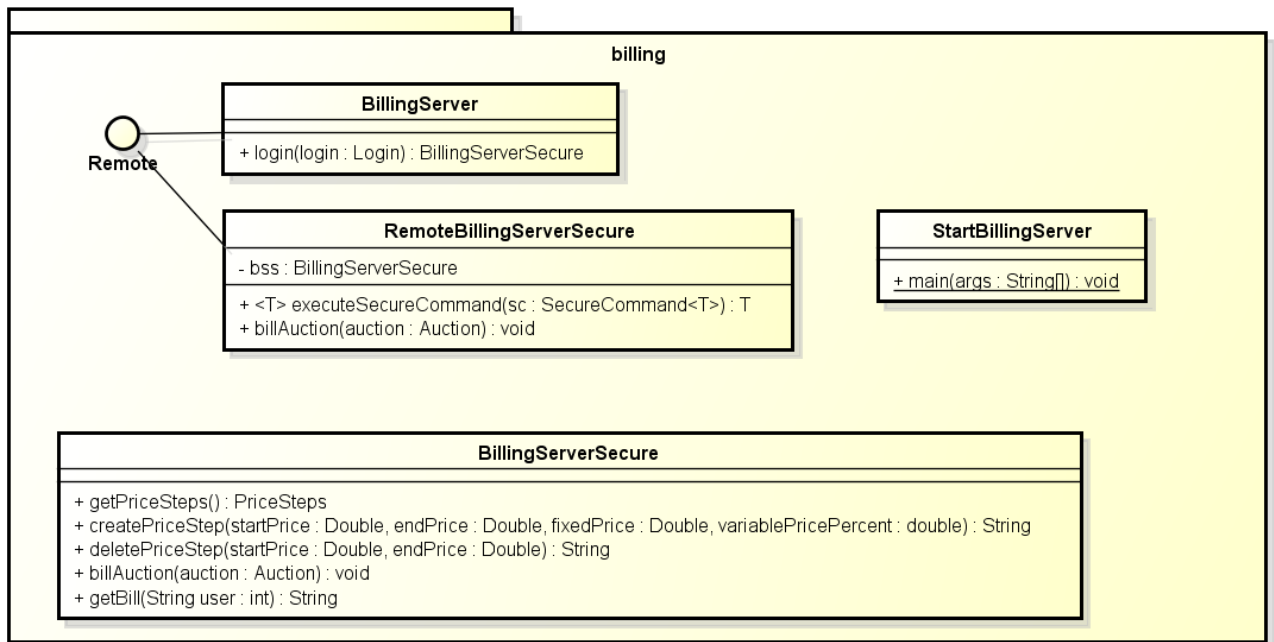


## AnalyticsChange with RMI

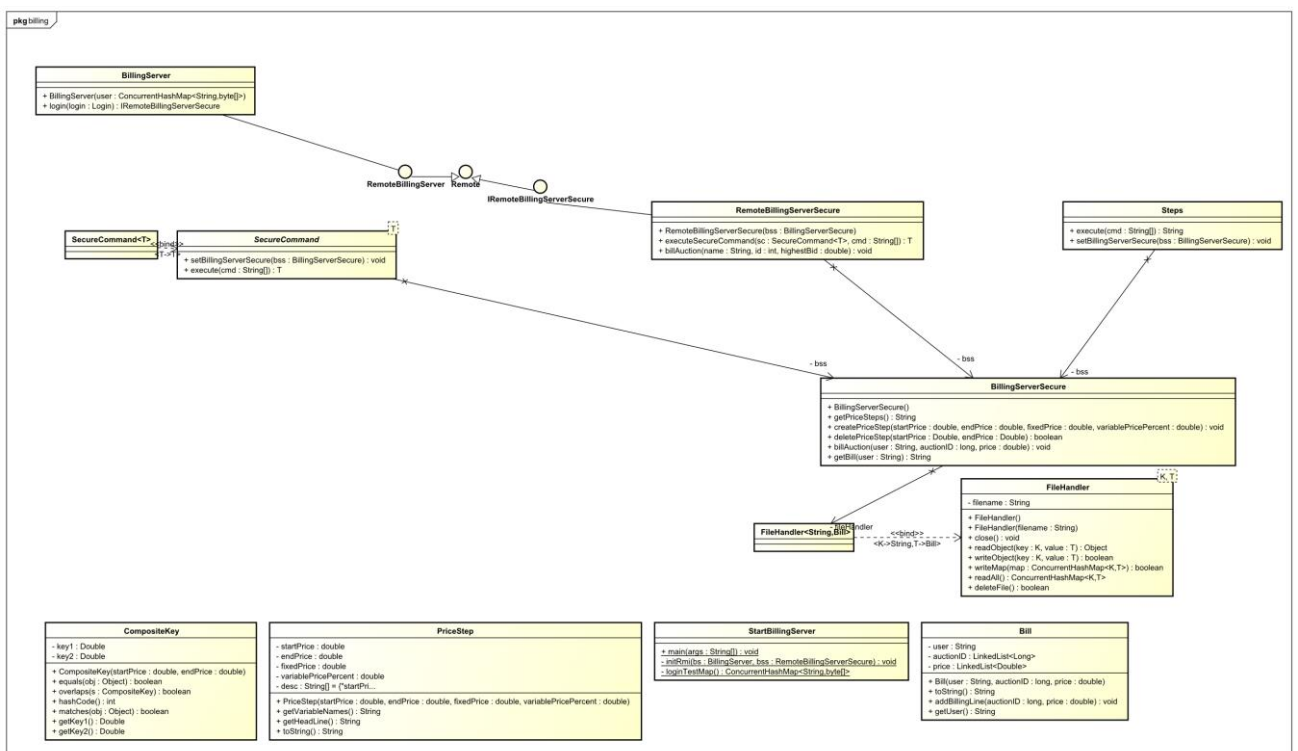




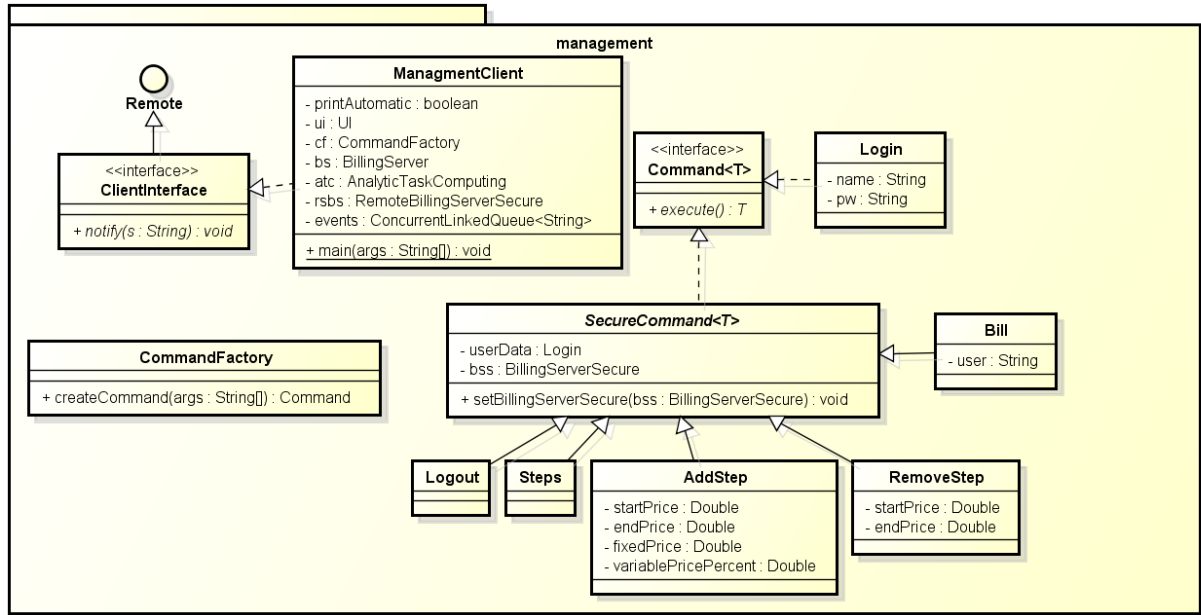
## Billing



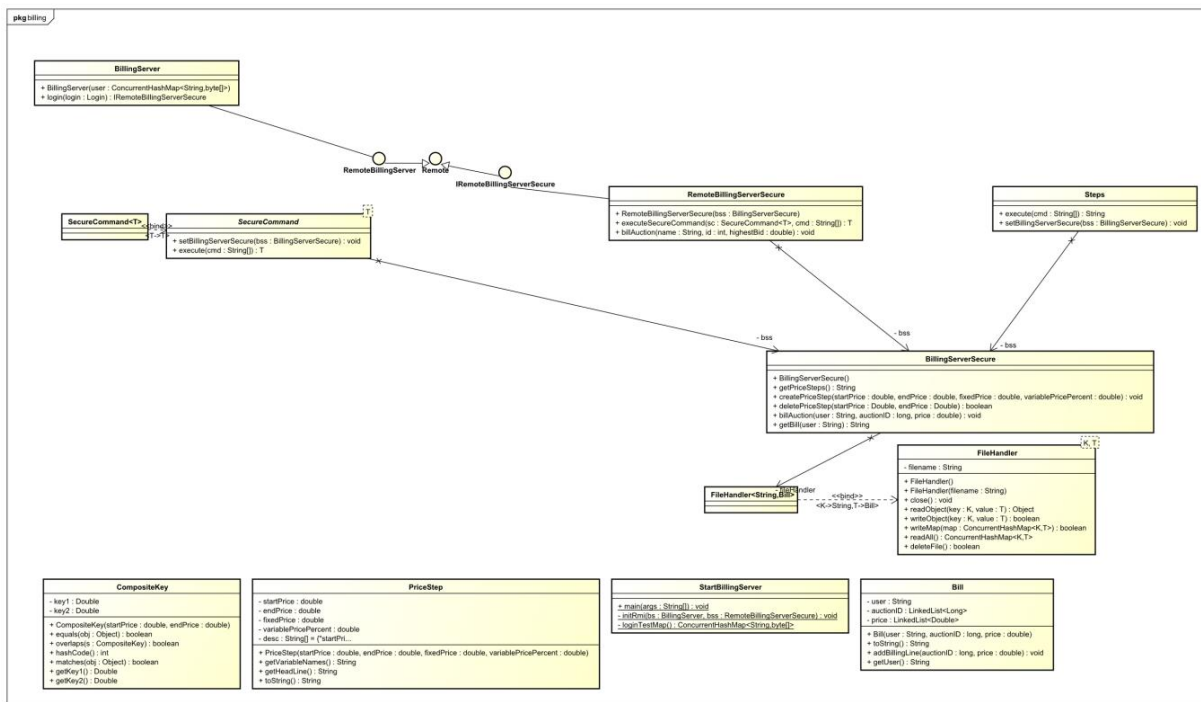
## BillingServer Change



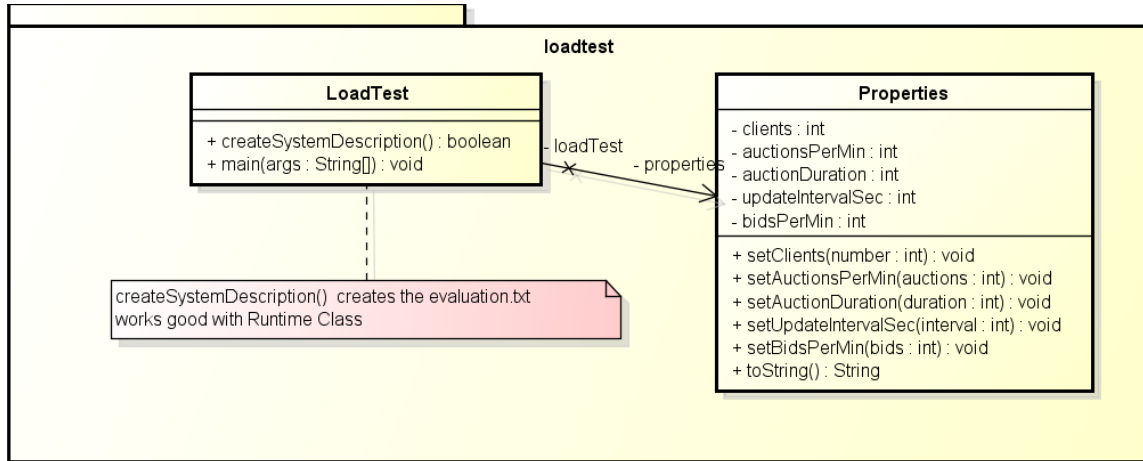
## Managementclient



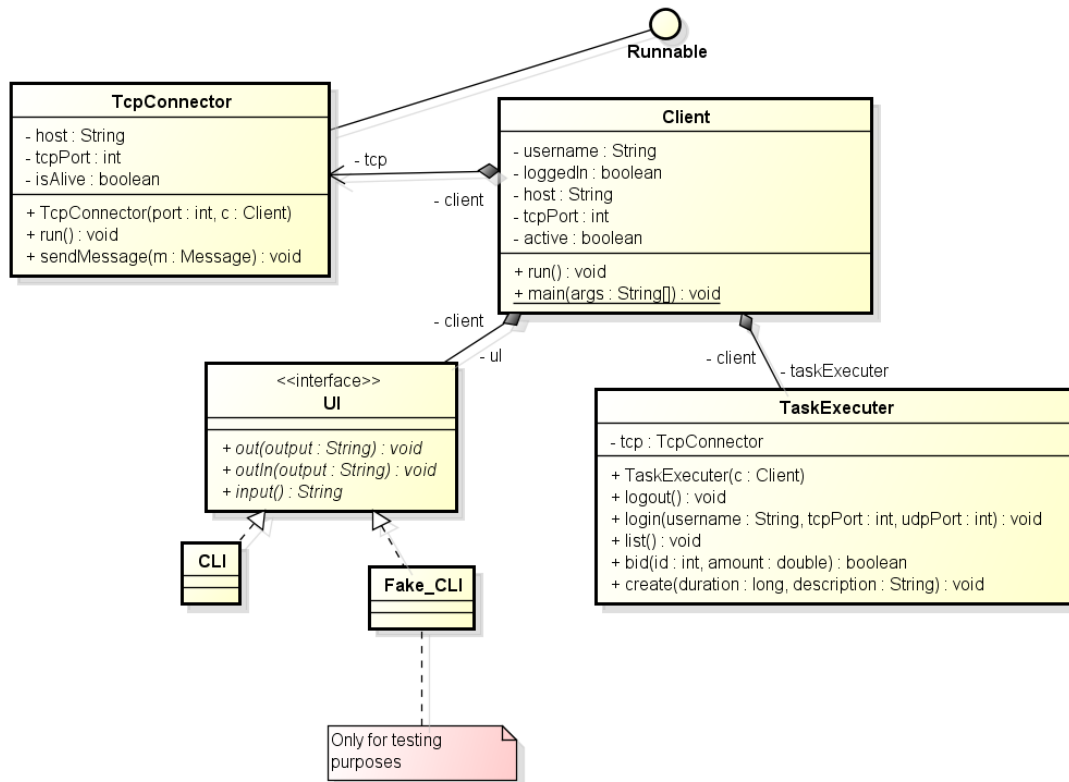
## ManagementClient



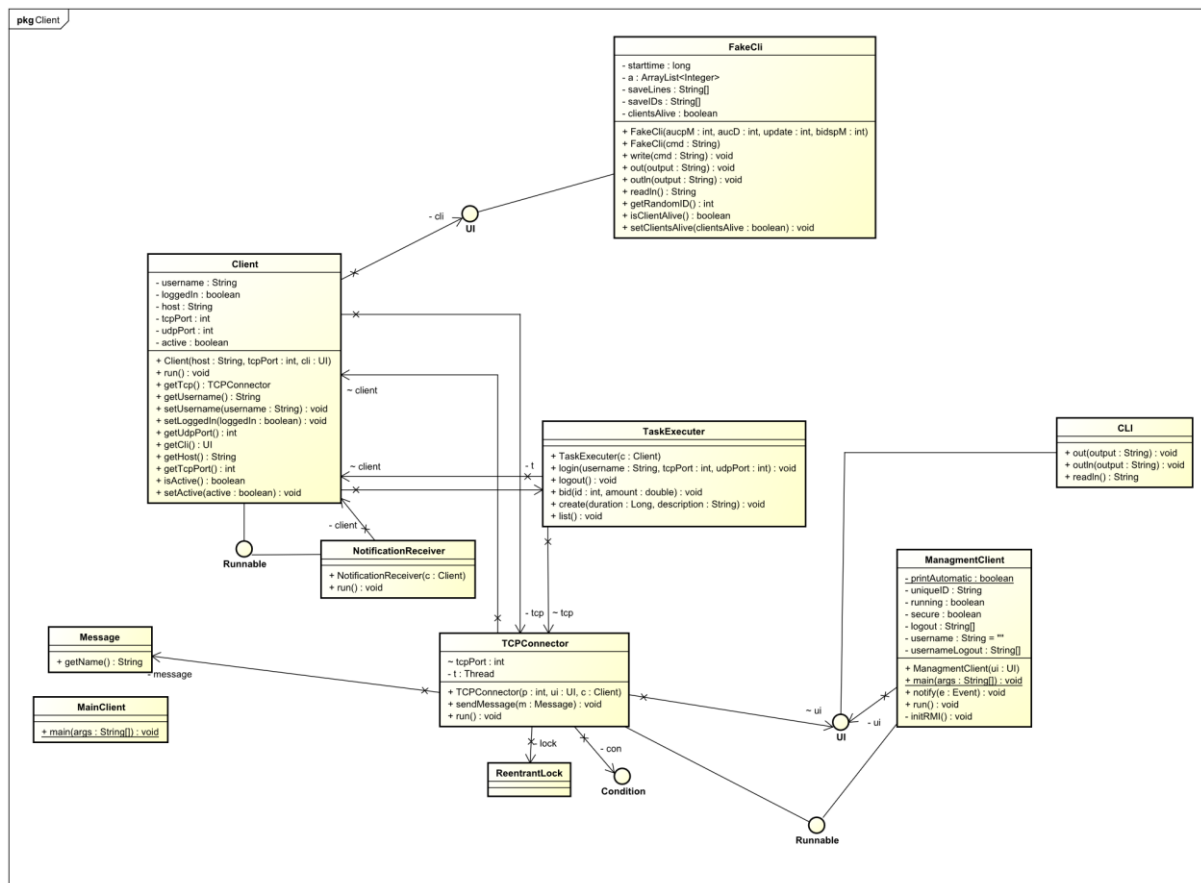
## Testing Component



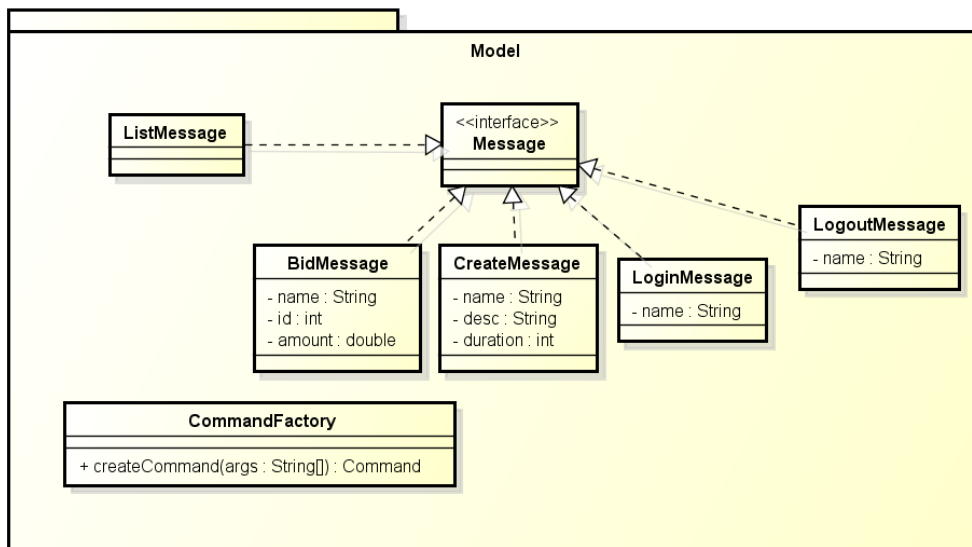
## Client



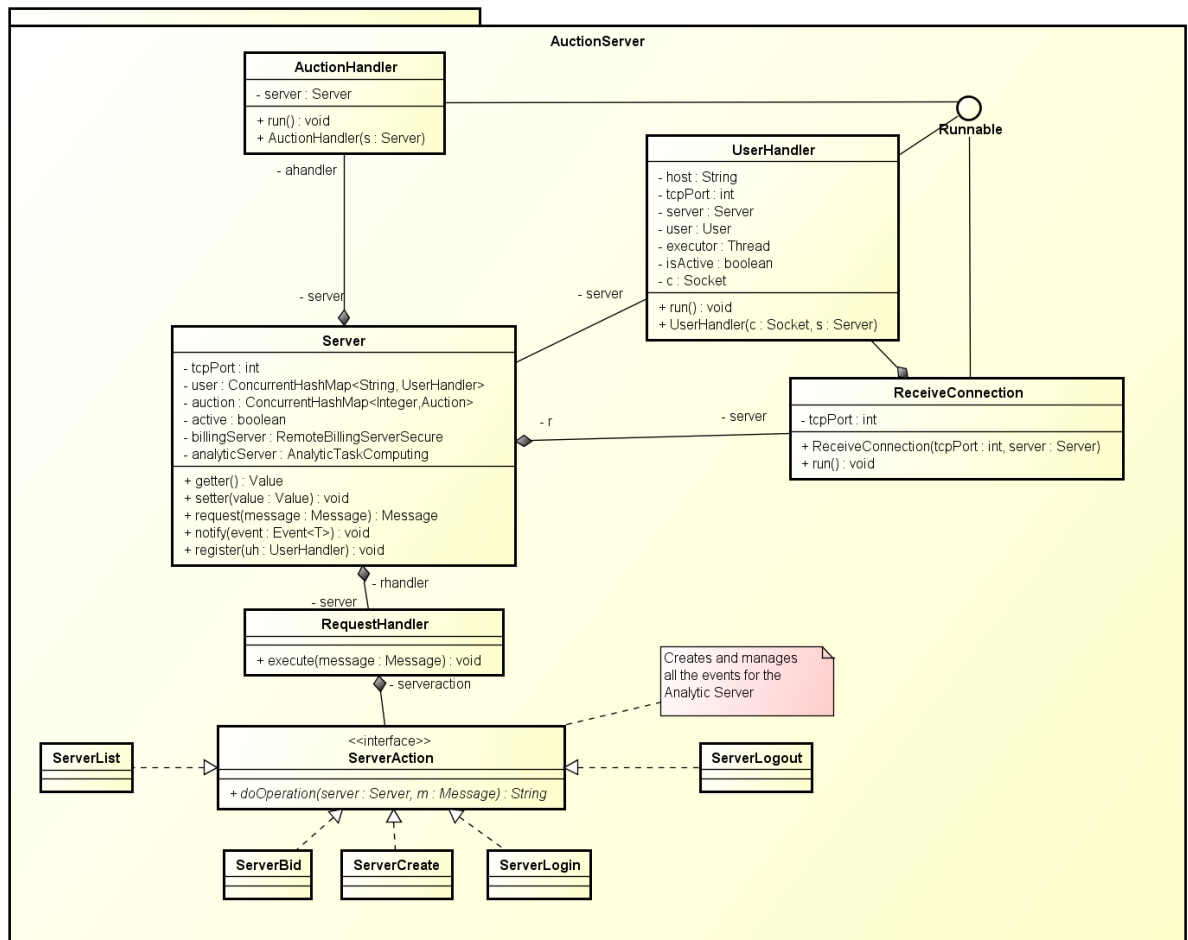
## Client Refactor



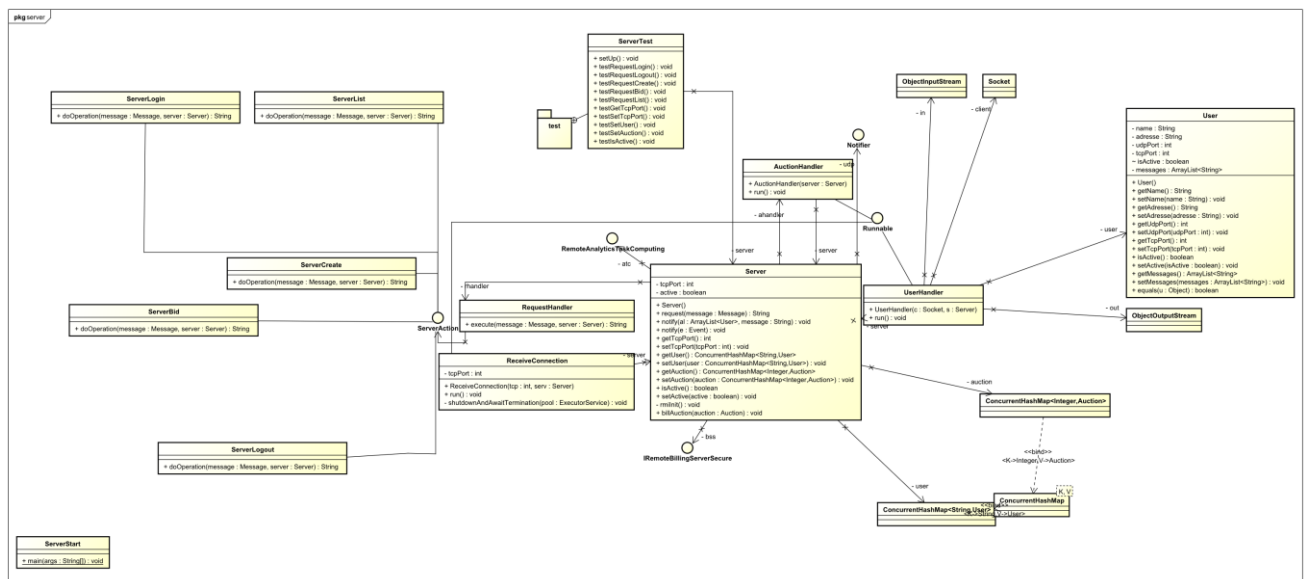
## Message-Model



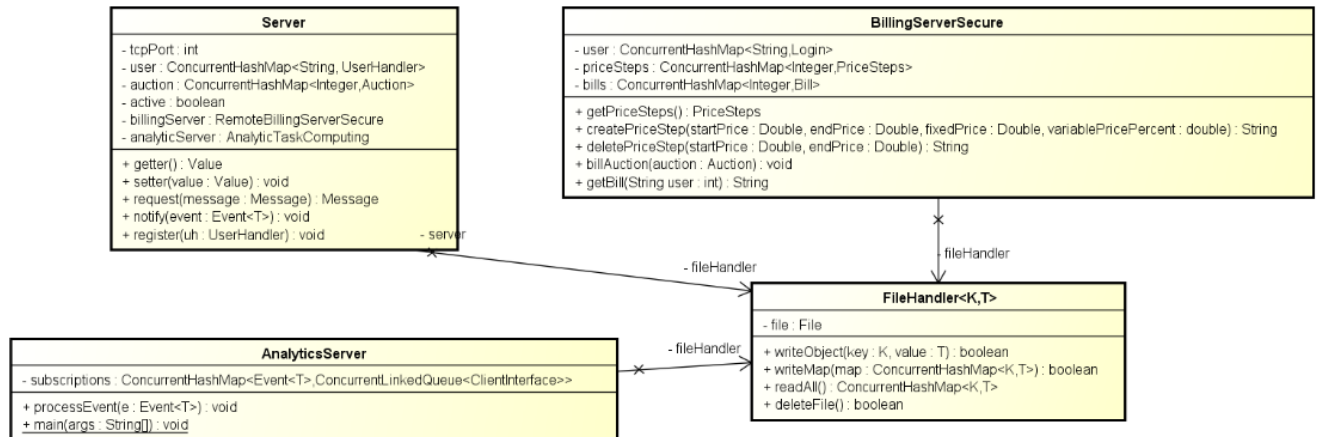
## Server



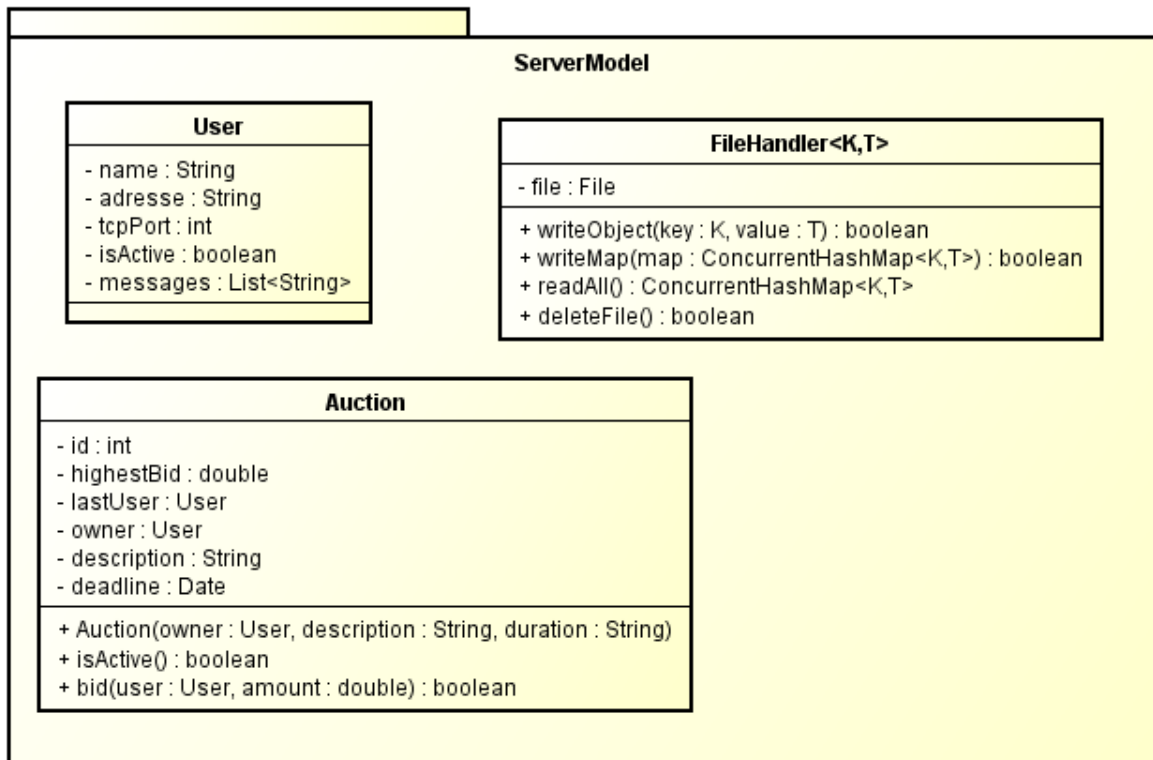
## Server Refactored



## FileHandler



## Model Server





## Zeitschätzungen und Arbeitsaufteilung

work package	Reichmann		Krepela		Lipovits		Tattyrek		Traxler	
	e.	r.	e.	r.	e.	r.	e.	r.	e.	r.
UML Klassendiagramm		1,50	2,00	2,00					2,00	4,00
UML Aktivitätsdiagramm					2,50	2,00				
UML Use-Case	1,00	0,50								
UML überprüfen							0,50	0,50		
Analytics-Server implementieren	6,00	4,00					3,00	0,50		
Billing-Server implementieren			6,00	4,00					4,00	1,50
Management-Client impl.					3,00	4,00	3,00	2,00		
Testing Component impl.					4,00	4,00				
Model-Klassen (Events, Bill, Steps)							1,50	1,00		
File-Persistence							2,00	2,50		
Refactoring old Source	2,00	1,50							2,00	0,50
RMI-Verbindungen implementieren	1,50	2,00							3,00	3,00
RMI-Verbindungen testen			1,00						1,00	
Analytics Unit testen	2,00	1,00					2,00	0,00		
Billing Unit testen			2,00	1,00						
Management-Client Unit testen					2,00					
Testing Component Unit testen					2,00					
Protokoll	2,00		2,00	0,10	1,00		1,00		1,00	
		3,00								
total	14,50	13,50	13,00	7,10	14,50	10,00	13,00	6,50	13,00	9,00
sum	68,00					46,10				

## 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, Benachrichtigungen etc)

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

UML-Klassendiagramm → Krepela, Traxler

Aktivitätsdiagramm → 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)
- Ausbessern 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 benötigt)

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 Komponenten 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.

Hallo liebes Team,

wie ihr wisst, müssen wir nächste Woche mit unserer Aufgabe fertig sein, da der liebe Herr Professor Borko uns die Tests abnimmt. Ich muss ihm nun noch seine Arbeitspakete senden (überlegen wir am Freitag)

Was heißt das für uns?

Jeder muss seinen Teil am Sonntag(! 16.02.14) fertig haben. Wie ich gesehen habe, fehlt euch allen nichtmehr viel und das sollte auf jeden Fall machbar sein. (AnalyseServer is am wenigsten weit, aber den muss sowieso ich auch machen).

Montag schau ich mir den Fortschritt an und teste euren Code.

Thomas: Bereite die Interfaces so vor, dass es funktioniert, wenn man die Teile zusammenfügt.

Nanak: Du übernimmst erstmal den FileHandler, ich überleg mir was zum AnalyseServer und sag dir wenn ich etwas brauch.

Es ist gut, wenn ihr vorher fertig seid, dann haben wir noch Zeit zum Debuggen.

Jeder schreibt bitte seine Dokumentation für seinen Part. Testing kann in der letzten Woche erfolgen, aber auch nicht bis irgendwann sondern rechtzeitig.

Das build.xml werde ich übernehmen

mfg  
Daniel

P.S.: Ich bitte um Rückmeldung und schreibt mich bitte an, wenn ihr Probleme jeglicher Art habt oder Hilfe braucht. Fragen bez. UML auch an Traxler/Krepela weiterleiten.

Liebes Team,

da bis gestern alle Eigenfunktionalitäten fertigzustellen waren, möchte ich bitte von euch bis morgen, dass die Funktionalitäten zusammengefügt werden.

Das bedeutet für Traxler, dass er die RMI-Objekte überall in die Registry schreiben muss, und für alle Server + MgmtClient verfügbar machen muss.

Bitte bis morgen 19:00.

Setzt euch morgen in NWSY/Deutsch zusammen und schaut, dass ihr dann etwaige Fehler gemeinsam Lösen könnt. Am Abend werde ich mir das ganze ansehen und testen.

mfg,  
Daniel