Opdracht - Veiling

Vak:
Internet technologie

Begeleidende docent:
Paul de Groot

Door:

Naam	***Email***	***Studentnummer***	***Klas***
Marco Jansen	Jansen.marco95@gmail.com	345864	EIN2vB
Daan Veldhof	Daanveldhof@gmail.com	357852	EIN2vB

Inhoudsopgave

Systeemdocumentatie

Beschrijving applicatie

Het systeem dat we gemaakt hebben is een veilingsysteem. Het veilingsysteem bestaat uit een server en clients. Wanneer de server aanstaat kunnen er verschillende clients zich verbinden met deze server. Een client moet dan inloggen en kan daarna de huidige veilingen bekijken, een bod uitbrengen of zelf een veiling beginnen.

De server houdt alle veilingen en accounts bij. Ook controleert de server wanneer een veiling is afgelopen. Als een veiling is afgelopen krijgt de hoogste bieder, indien aanwezig, een bericht dat hij/zij een veiling heeft gewonnen. In dat bericht staat ook informatie over welke veiling hij/zij heeft gewonnen. Wanneer een veiling is afgelopen kan hier nog wel informatie over opgevraagd worden, maar kan er geen bod meer gedaan worden.

Opbouw applicatie

Onze applicatie bestaat uit vier packages.

1. Server
2. Client
3. Model
4. Test

In de server package staan de volgende classes :

- Server.java
- ClientListener.java
- AuctionWatcher.java

Wanneer de Server.java uitgevoerd wordt, wordt de server opgestart. Voor elke client die zich aanmeld wordt er een ClientListener gestart. De ClientListener is een thread die constant luistert naar de bijbehorende client zonder dat het programma stil wordt gezet. Ook wordt er eenmalig een AuctionWatcher gestart. Dit is een thread die om de seconde kijkt of er veilingen zijn afgelopen.

In de client package staan de volgende classes :

- Client.java
- ServerListener.java

Wanneer de Client.java uitgevoerd wordt, wordt er een client aangemaakt. De client probeert zich dan met de server te verbinden. Voor elke client wordt er een ServerListener gestart. De ServerListener zorgt ervoor dat de client in een andere thread constant naar de server kan luisteren zonder dat het programma stil wordt gezet.

In de model package staan de volgende classes :

- Model.java
- Auction.java
- Account.java

Wanneer de server opgestart wordt, wordt er een model aangemaakt. In het model wordt een lijst met accounts, active auctions en ended auctions bijgehouden. Elke account heeft een username, password en een socket. De socket wordt bijgehouden zodat de server weet welke socket er bij welk account hoort en hier berichten naartoe kan sturen. Elke Auction heeft een id, item, description en expirationdate. Ook wordt er bij een auction bijgehouden wat het hoogste bod is en wie de hoogste bieder is.

In de test package staan de volgende classes :

- ModelTest.java
- AuctionTest.java
- AccountTest.java
- ServerTest.java

Al deze classes worden gebruikt als unit test om onze classes te testen. Bij onze unit tests maken we gebruikt van JUnit 4. In de naam van de test wordt verwezen naar welke class er wordt getest.

Verantwoording keuzes

TCP/UDP

We hebben gekozen om TCP te gebruiken. Dit omdat UDP niet garandeert dat alle pakketten aankomen en TCP garandeert dit wel. In een veiling systeem het cruciaal dat alle biedingen aankomen.

AuctionWatcher

Onze AuctionWatcher is een thread. Hiervoor hebben we gekozen omdat een thread op de achtergrond uitgevoerd kan worden en het programma dus niet stil zet.

ServerListener

Onze ServerListener is een thread. Hiervoor hebben we gekozen omdat er geluisterd moet worden naar de server en ook naar de gebruiker en dit kan niet allebei in dezelfde thread worden uitgevoerd.

ClientListener

Onze ClientListener is een thread. Hiervoor hebben we gekozen omdat er geluisterd moet worden naar meerdere clients en er ook geluisterd moet worden naar de server. Doordat de ClientListener een thread is hebben we er geen last van dat er gewacht moet worden op invoer aangezien het programma hierdoor niet stilgezet wordt. Ook is het hierdoor mogelijk om naar meerdere clients tegelijk te luisteren.

Feedback

We hebben ervoor gekozen om er voor te zorgen dat de gebruiker feedback krijgt wanneer er iets fout gaat. Dit zodat de gebruiker weet wat er aan de hand is. Onze feedback ziet er als volgt uit :

Situatie	**Feedback**
Er kan geen verbinding met de server gemaakt worden.	Connection refused.
Er is verbinding gemaakt met de server.	Connection is working.
De server gaat na het verbinden plotseling uit.	Server is not responding.
Er is een fout (niet bestaand) commando ingevoerd.	Wrong input, please try again.
Tekst invullen i.p.v. een nummer als dat gevraagd is.	Not a number, try again.
Er zijn geen actieve veilingen en de gebruiker vraagt om een lijst met veilingen.	There are no active auctions.
Een gebruiker zoekt op een keyword wat in geen een veiling voorkomt.	There were no auctions found with this keyword
Een gebruiker vraagt een auction op met een id wat niet bestaat.	No auction found with this id OF Auction doesn't exist (ligt aan het commando)
Er wordt een bod uitgebracht wat lager is dan het hoogste bod.	Bid wasn't high enough
Er is een fout opgetreden	Error! (+ wat er mis is gegaan)

Protocolbeschrijving

Transportprotocol

Veilingsysteem maakt gebruik van het TCP transportprotocol. Met het TCP protocol wordt er een verbinding opgezet en pas gesloten als de server of de client ervoor kiest om de verbinding te verbreken. Het TCP protocol verzekerd op zekere hoogte ook dat alle pakketten overkomen. Als er bij dit systeem gebruik werd gemaakt van UDP was er geen garantie dat de communicatie goed verliep.

Berichten

Hier onder zijn alle mogelijke requests en responses te vinden van het protocol. Wat er achter request en response staat is een normale regel tekst. Op de plek van [ ] haken moet een woord, regel of nummer komen te staan die slaat op de tekst binnenin de blokhaken. Dus als er [AuctionId] staat moet er een auction id komen te staan, bijvoorbeeld 6. Elke keer als er drie puntjes staan (...) wordt het bericht vervolgd met hetzelfde als wat er voor staat. Dit deel van het bericht kan dus herhaald worden

Alle veilingen ophalen

Client request

getAuctions

Dit is het commando aan de server om alle veilingen terug te sturen.

Server response

getAuctions [auctionID],[name],[description]<>[auctionID],[itemName],[itemDescription]<>...

De server stuurt als response een lijst met veilingen terug. Elke veiling heeft een ID, naam en beschrijving.

Veilingen zoeken

Client request

searchAuction [keyword]

Dit is het commando aan de server om veilingen waar het keyword inzit terug te sturen

Server response

searchAuctions [ID],[name],[description],[highestBid]<>[ID],[name],[description],[highestBid]<>

De server stuurt als response een lijst met veilingen terug. Elke veiling heeft een ID, naam, beschrijving en het hoogste bod.

error No keyword

De server stuurt een error bericht als er geen keyword is meegegeven.

Informatie van één veiling ophalen

Client request

getAuctionInfo [auctionID]

Dit is het commando aan de server om informatie van een veiling op te sturen

Server response

getAuctionInfo [name],[description],[highestBid],[ended]

De server stuurt als response een veiling terug. Deze veiling heeft een naam, beschrijving, hoogste bod en een boolean (true of false) of de veiling al beëindigd is of niet.

error auction [auctionID] doesn't exist

Deze response wordt gestuurd als er geen veiling is gevonden met het opgegeven ID

error No valid auction id

Deze response wordt gestuurd als de opgegeven ID niet juist is.

Veiling toevoegen

Client request

addAuction [name]<>[description]<>[duration]<>[*startprice]

Dit is het commando aan de server om een veiling toe te voegen met de ingegeven parameters. De **startprice* is optioneel.

Server response

addAuction true

Deze response wordt gestuurd als het toevoegen van de veiling is gelukt

error no valid parameter for addAuction

Deze response wordt gestuurd als er een parameter niet goed is ingevoerd of helemaal niet is ingevoerd.

Bod plaatsen

Client request

doOffer [auctionID]<>[price]

Dit is het commando aan de server om een bod te doen op een veiling.

Server response

doOffer true

De response van de server als het bod geplaats is

doOffer false

De response van de server als het bod niet is geplaatst, omdat het bod te laag was

error No valid price given

De response van de server als het bod niet goed was

error auction [auctionID] has ended

De response van de server als de veiling al geëindigd is

error Auction doesn't exist

De response van de server als de veiling niet bestaat

error No valid arguments were given

De response van de server als het commando fout is ingevoerd

Hoogste bod opvragen

Client request

highestOffer [auctionID]

Dit is het commando aan de server om het hoogste bod van de ingevoerde veiling terug te geven

Server response

highestOffer [highestBid]

De response van de server met het hoogste bod

error auction [auctionID] has ended

De response van de server als de veiling geëindigd is.

error auction [auctionID] doesn't exist

De response van de server als de veiling niet bestaat

error No valid auction id

De response van de server als het id niet goed is.

Opvragen hoelang de veiling nog duurt

Client request

auctionEnds [auctionID]

Dit is het commando aan de server om te vragen hoelang de veiling nog duurt

Server response

auctionEnds [remainingTime]

De response van de server. De resterende tijd is in milliseconden

error auction [auctionID] doesn't exist

De response van de server als de veiling niet bestaat

error no valid auction id

De response van de server als het opgegeven id niet goed is

Speciale berichten

Server response

error no valid command

Dit is de response van de server als een client een request heeft gedaan die niet bestaat.

Server announcement

winner [auctionID]

Dit is een announcement die de server geeft aan een client als een veiling is geëindigd en de client had het hoogste bod.