Opdracht: Studiebeurs

Verwachte studiebelasting: 15 uur per persoon

Doel

In deze opdracht werk je in **tweetallen** om een netwerkprotocol te ontwerpen en een proof-of-concept te implementeren. Aan het eind lever je in en wordt beoordeeld:

- Een document met je protocol-ontwerp, waarbij de diverse protocolbeschrijvingen en RFC's op Internet als voorbeeld dienen[1]. Dit document bevat tenminste:
  - Een state diagram van zowel server als client
  - Een beschrijving van de berichten die kunnen worden verstuurd
  - De reactie van het systeem (zowel client als server) bij geldige én ongeldige berichten
  - Beargumentering van de belangrijkste ontwerpbeslissingen, bijvoorbeeld:
    - Welk transport-protocol gebruik je (UDP of TCP)?
    - Kies je voor tekstuele of binaire berichtuitwisseling en waarom?
    - Hoe ga je om met gesloten verbindingen?
- Een tweetal applicaties (client en server) dat het ontworpen protocol implementeert
- Eventuele (unit) testcode
- Een testplan en testrapport
- Een beknopte handleiding van de opgeleverde software: welke applicaties dienen gestart te worden en hoe worden de commando's aan de server gegeven?

Casus

In tijden van economische crisis wordt het steeds moeilijker voor organisaties om het hoofd boven water te houden. Zo ook voor de studieverenigingen van Saxion. Daarom hebben ze samen besloten om de beurs op te gaan. Alle verenigingen van Saxion gaan aandelen uitgeven en deze worden verhandeld op een (client/server)-beurssysteem. Jij bent gevraagd om daarvoor het netwerkprotocol te ontwerpen.

Alle studenten, medewerkers en studieverenigingen van Saxion kunnen een username en password krijgen waarmee ze in kunnen loggen op het beurssysteem. Aan elk account is een saldo gekoppeld: door geld naar Saxion te storten kan geld op je beurs-account worden bijgeschreven. Met het geld op het account kunnen binnen het beurssysteem aandelen in een studievereniging worden aangeschaft.

Op de server van het systeem wordt de administratie bijgehouden van alle gebruikers en wie welke aandelen bezit. Gebruikers van het systeem kunnen dan clients gebruiken om met de server te communiceren.

[1] Een redelijk leesbare protocol specificatie is bijvoorbeeld http://stomp.github.io/stomp-specification-1.1.html

Elke vereniging heeft 5000 aandelen uitgebracht. Deze aandelen zijn initieel in handen van de desbetreffende vereniging, maar worden te koop aangeboden voor € 5,00 per stuk. Neem in je implementatie minimaal 3 studieverenigingen op. (Er zijn er bijna 20 op Saxion).

Als een gebruiker aandelen wil verkopen, kan hij dit bij het beurssysteem aangeven. Hij maakt dan een aanbieding van een aantal aandelen en geeft de prijs aan waarvoor hij ze wil verkopen. Bijvoorbeeld: gebruiker *Pietje1996* kan aangeven 50 aandelen Syntaxis van de hand te willen doen voor € 2,50 per stuk. Aandelen die je niet bezit, kan je ook niet te koop aanbieden.

Gebruikers die aandelen willen kopen, kunnen een lijst opvragen met de huidige aanbiedingen van een bepaald type aandeel. Bijvoorbeeld, als gebruiker kan je de aanbiedingen van het aantal LiNK opvragen en als antwoord de volgende lijst krijgen:

Aanbieder	Aantal	Prijs
RuudGr	43	€ 2,50
WimB	10	€ 2,75
HenkieM	65	€ 2,40

Als daar een aanbieding bij zit die interessant lijkt, kunnen gebruikers aangeven van die aanbieding een aantal, of alle aandelen te willen kopen. De gebruiker moet hiervoor natuurlijk wel voldoende saldo hebben en de aandelen moeten inmiddels niet al zijn verkocht! De transactie wordt vervolgens direct verwerkt.

De bovenstaande casus is waarschijnlijk niet helemaal compleet. Mocht je iets tegenkomen wat niet door de tekst gedekt wordt, denk dan eerst na welke keuzes je zou kunnen maken. Maak dan een overwogen beslissing en neem de redenen op in je documentatie. Vanzelfsprekend is een gebruiksvriendelijk en robuust systeem belangrijker dan een simpeler implementatie. Bij twijfel kan je natuurlijk ook je docent vragen.

Programma

Aan de programmatuur zijn niet echt eisen. Houd in je achterhoofd dat het gaat om de communicatie.

Voor de dataopslag kan dus worden volstaan met een simpele in-memory oplossing. Een SQL-database of efficiënte datastructuur is niet noodzakelijk (maar mag wel).

Dit geldt ook voor de user interface. Dit mag een simpele tekstinterface zijn in Java SE of een fancy app in Android. De interface heeft geen (grote) invloed op je cijfer, de algemene verzorgdheid (logische structuur, heldere naamgeving, tests en Javadoc) wel.

Als je deze opdracht in je eentje doet mag je de functionaliteit zoals beschreven in de casus minimaliseren, eisen aan verslaglegging en nette programmatuur blijven natuurlijk.

Veel succes!