Verslag Connect4 project

Michael Koopman s1401335 en Sven Konings s1534130

# Inleiding

In deze module hebben wij ervaring opgedaan in Java. In dit project hebben wij die kennis in de praktijk toegepast. Dit project bestond uit het maken van een Client en een Server voor het Connect4 spel, waarbij aan de hand van een protocol, spelletjes gespeeld kunnen worden met andere mensen van onze practicumgroep met een andere implementatie dan de onze. Dit verslag zal onder andere inzicht geven in ons design, hoe wij de klassen geïmplementeerd hebben, en hoe onze tests werken.

# Discussie van het algehele design

## Klassediagrammen

### Client/Server

### Game

## Systematisch overzicht van welk van de eisen is geïmplementeerd in welke klassen

|  |  |
| --- | --- |
| **EIS** | **GEÏMPLEMENTEERD IN:** |
| **Server** |  |
| Als de server is gestart, moet er een poortnummer ingevoerd worden waarnaar de server zal luisteren. | ServerGUI. Bij het opstarten van de ServerGUI kun je een poort invoeren, en daarna kun je klikken op “Start listening”. Na het klikken op “Start listening”, kun je de poort niet meer veranderen (tenzij de poort al gebruikt wordt door een ander programma). |
| Als het poortnummer al gebruikt wordt, wordt een geschikte foutmelding gegeven, en kan een nieuw poortnummer ingevoerd worden. | ServerGUI. Als je een poortnummer invoert, die al in gebruik is, dan krijg je het bericht: “Error listening on port <port>, please select a different one”, en wordt het invoerveld voor port weer bewerkbaar. |
| Een Server moet meerdere instanties van het spel ondersteunen, die tegelijkertijd door verschillende Clients gestuurd worden. | ClientHandler. 2 ClientHandlers krijgen hetzelfde bord, en die regelen de Game verder. Als er een move gecheckt wordt, is alleen de ClientHandler van degene die de move wil doen bezig. |
| De TUI zorgt ervoor alle communicatie naar System.out geschreven. | ServerGUI. Wij hebben onze implementatie gebaseerd op de Multi-Client chat uit week 7, en daarbij hebben we de ServerGUI klasse gebruikt, die daarbij zat. Hierdoor hebben wij geen TUI voor de Server. Alle berichten worden geschreven naar het berichtenvak in de GUI, en je kunt scrollen door dat vak. Hierdoor is de functionaliteit hetzelfde. |
| De Server respecteert het protocol zoals afgesproken met de tutorialgroep tijdens de projectsessie in week 7, oftewel de Server moet kunnen communiceren met alle andere Clients. | Features, Server en (vooral) ClientHandler. Er zijn constanten gedefinieerd in Server en Features, zoals die in het protocol staan, en die worden door zowel Server als ClientHandler continu gebruikt. |
| **Client** |  |
| De Client moet een gebruiksvriendelijke TUI hebben, die verschillende opties aanbiedt aan de gebruiker, zoals het invoeren van een IP-adres en poort om een Game van de Server op te vragen. | ClientTUI en Client. Zodra er een Client object wordt gemaakt, wordt aan de TUI om een IP-adres en een poort gevraagd. Als er een verbinding kan worden opgezet, dan vraagt de Client de TUI om een naam. Tijdens het vragen van de naam kan ook aangegeven worden, dat je een ComputerPlayer met een Strategy voor je wilt laten spelen. Kan er geen verbinding gemaakt worden, dan wordt er opnieuw om een IP en een poort gevraagd. Verder werkt onze Tutorialgroep met een lobby en invite systeem om een Game op te vragen. |
| De Client moet HumanPlayers ondersteunen, en ComputerPlayers met een beetje kunstmatige intelligentie. | ClientTUI en Client. Bij het opgeven van je naam, kun je kiezen of je zelf wilt spelen, of de computer voor je wilt laten spelen. We hebben onze eigen versie van het MinMax-algoritme geïmplementeerd, aan de hand van een uitleg over hoe het algoritme werkt, die we vonden op het internet. |
| De bedenktijd van de ComputerPlayer moet een parameter zijn, die veranderd kan worden in de ClientTUI. | ClientTUI en Client. Door het commando DIFFICULTY te gebruiken kun je de diepte (het aantal zetten dat het algoritme vooruit denkt) van de MinMaxStrategy aanpassen. De overige Strategies hebben een verwaarloosbare bedenktijd, dus hiervoor is het niet nodig, om een aparte parameter te maken. |
| De Client moet een hintfunctionaliteit hebben, die een HumanPlayer een mogelijke move laat zien, voorgesteld door een ComputerPlayer. De move moet alleen voorgesteld worden, en niet automatisch gedaan worden. | ClientTUI, Client, ComputerPlayer en MinMaxStrategy. Als het HINT commando worst ingevoerd in de ClientTUI, wordt dat aan de Client doorgegeven, en de Client laat een nieuwe ComputerPlayer met een MinMaxStrategy een move bepalen op een kopie van het huidige Board. Deze wordt vervolgens voorgesteld aan de HumanPlayer, maar niet daadwerkelijk gedaan. |
| Nadat het spel is afgelopen, moet de speler een nieuw spel kunnen beginnen. | Server, ClientHandler en Client. Als het spel is afgelopen, wordt er een game end pakket gestuurd, en gaat de Client weer naar de lobby. Vanuit de lobby kan hij dan weer iemand inviten of een invite van iemand anders accepteren. |
| Als een speler het spel afsluit voordat het is afgelopen, de UI sluit, of de client crasht, moeten de andere spelers daarover geïnformeerd worden. In dit geval moeten de andere spelers weer kunnen registreren met de Server om een weer te kunnen spelen. | ClientTUI, Client, Server en ClientHandler. Als de Client het QUIT commando gebruikt, dan wordt er nog een message gestuurd naar de Server, met dat de Client gequit heeft, anders wordt er door de Server aan de andere Clients gezegd dat die Client gedisconnect is. |
| Een Server kan op ieder moment disconnecten. De Client moet hier op een goede manier mee omgaan. | Client. De Client sluit af op het moment dat de server disconnect. |
| De Client respecteert het protocol zoals afgesproken met de tutorialgroep tijdens de projectsessie in week 7, oftewel de Client moet kunnen communiceren met alle andere Clients. | Client en Features. Er zijn constanten gedefinieerd in Client en Features, zoals die in het protocol staan, en die worden door Client continu gebruikt. |
| **Global** |  |
| De Client en de Server moeten altijd in dezelfde game state zijn. | Als de game state van de server verandert, stuurt de server een MOVE OK pakket met player nummers. Daarmee kunnen de Clients bijhouden wie er aan de beurt is, en de move op het Board doen. |
| De game mag niet gekopieerd zijn van het internet. | Alle klassen zijn door onszelf geschreven, gebaseerd op de Multi-Client chat van week 7. |

## Het gebruik van het Observer en het Model-View-Controller patroon.

### Observer

### Model-View-Controller

## Opslag van data en communicatieprotocollen

Ons leaderboard wordt opgeslagen in een txt-file. Verder slaan wij lokaal, behalve de java-bestanden natuurlijk, geen data op. Informatie over de communicatie tussen de Client en de Server kunt u vinden in het protocol van onze tutorialgroep, TI-2.

# Discussie per klasse

## Client.java

### Rol van de klasse in het systeem

Client.java regelt alles client-side, van het opvangen van servercommando’s, tot het sturen van commando’s en het bijhouden van het bord.

### De verantwoordelijkheden van deze klasse

Deze klasse regelt het opvangen en verwerken van de volgende servercommando’s:

* ACCEPT\_CONNECT: Client geeft door aan de TUI dat er een succesvolle verbinding is opgezet met de server, en welke features de server heeft. Ook wordt de boolean isConnected op true gezet.
* LOBBY: Client geeft de spelers in de lobby door aan ClientTUI.
* INVITE: Client slaat een nieuwe invite op, en geeft aan de TUI door dat de Client geinvite is.
* DECLINE\_INVITE: Client verwijdert de invite van de persoon, en geeft aan de TUI door dat de invite gedeclined is.
* GAME\_START: Zet myNumber op nummer van de speler, zoals dat op de server bekend is, en geeft aan de TUI door, dat er een nieuwe game is begonnen tussen jou en de andere speler.
* GAME\_END: Zet het board op null, en geeft aan de TUI door, wie heeft gewonnen (of dat het gelijkspel was). Mocht de server dit niet hebben doorgegeven, dan wordt doorgegeven dat de game is geëindigd vanwege de reden die de Server meegaf.
* REQUEST\_MOVE: Checkt of de speler een HumanPlayer is, en zo ja, geeft door aan de TUI dat de speler een move moet doen, en zo niet, laat de ComputerPlayer een move maken.
* MOVE\_OK: Probeert de move te doen op het board. Mocht dit niet lukken, dan wordt het bord opgevraagd.
* ERROR: Geeft door aan de TUI dat er een error is gebeurd.
* BOARD: Verandert het Board naar het bord, dat de Server net doorgaf.
* CHAT: Geeft een chatbericht door aan de TUI.
* LEADERBOARD: Zorgt ervoor dat het leaderboard in een goede vorm wordt doorgegeven aan de TUI.
* PONG: Geeft door aan de TUI dat de server een PONG bericht stuurde.
* Alle andere commando’s: Geef door aan de server: Unknown command.

Verder regelt deze klasse het verwerken van de volgende clientcommando’s: