**PROGRAMMING PROJECT**

SOFTWARE SYSTEMS

Lieke Hamelers (s1741640)

Amber Altenburg (s1477226)

Januari 2016

Universiteit Twente

Inhoudsopgave

[1 Discussion of the Overall Design 3](#_Toc473538013)

[1.1 Class Diagrams 3](#_Toc473538014)

[1.2 Functional Requirements 4](#_Toc473538015)

[1.2.1 Functional requirements voor de server 4](#_Toc473538016)

[1.3 Model-View-Controller pattern 5](#_Toc473538017)

[1.3.1 Model 5](#_Toc473538018)

[1.3.2 View 5](#_Toc473538019)

[1.3.3 Controller 5](#_Toc473538020)

[1.4 Data Storage and Communication 6](#_Toc473538021)

[2 Discussion per Class 7](#_Toc473538022)

[2.1 Board 7](#_Toc473538023)

[2.2 Color 7](#_Toc473538024)

[2.3 Field 7](#_Toc473538025)

[2.4 Player 7](#_Toc473538026)

[2.5 Game 8](#_Toc473538027)

[2.6 ThreeDRow 8](#_Toc473538028)

[3 Test report 10](#_Toc473538029)

[3.1 Unit Testing 10](#_Toc473538030)

[3.1.1 BoardTest 10](#_Toc473538031)

[3.1.2 FieldTest 11](#_Toc473538032)

[3.1.3 GameTest 11](#_Toc473538033)

[3.1.4 PlayerTest 11](#_Toc473538034)

[3.2 System testing 13](#_Toc473538035)

[4 Metrics report 14](#_Toc473538036)

[5 Reflection on Planning 15](#_Toc473538037)

[5.1 Planning 15](#_Toc473538038)

# 1 Discussion of the Overall Design

TODO: Class diagram invoegen

## 1.1 Class Diagrams

## 1.2 Functional Requirements

### 1.2.1 Functional requirements voor de server

|  |  |
| --- | --- |
| **Functional requirement** | **Geïmplementeerd door:** |
| 1. Wanneer de server wordt gestart, moet er een poort nummer ingevuld worden waarnaar de server zal luisteren. |  |
| 1. Als het poortnummer al in gebruik is, wordt er een passende foutmelding teruggegeven en kan er een nieuw poortnummer worden ingevoerd. |  |
| 1. Een server moet meerdere exemplaren van het spel, die tegelijkertijd door verschillende clients worden gespeeld, kunnen ondersteunen. |  |
| 1. De server heeft een TUI die ervoor zorgt dat alle communicatie berichten naar *System.out* worden geschreven. |  |
| 1. De server moet het protocol zoals gedefinieerd voor de tutorial groep tijdens de projectsessie in week 7 respecteren, dat wil zeggen, de server moet kunnen communiceren met alle andere cliënten uit de tutorial groep. |  |

## 1.3 Model-View-Controller pattern

### 1.3.1 Model

De volgende classes vormen het model: Board, Color, ComputerPlayer, Field, HumanPlayer en Player.

De model classes representeren de gegevens van het Connect Four 3D programma. Zij beheren de gegevens en voeren alle transformaties van die gegevens uit. Verder zijn deze classes niet op de hoogte van de controllers en views; ze bevatten geen verwijzingen naar beiden.

### 1.3.2 View

De volgende class vormt de view: TUIView.

De bovenstaande class beheert de visuele weergave van de gegevens die door het model worden gerepresenteerd. TUIView zorgt ervoor dat er data zichtbaar wordt voor de spelers van het Connect Four 3D programma.

### 1.3.3 Controller

De volgende classes vormen de controller: Game en ThreeDRow.

De controller classes combineren de gegevens uit het model met de view. Daarnaast reageren de controllers op gebeurtenissen die meestal het gevolg zijn van handelingen van een gebruiker, bijvoorbeeld het doen van een zet.

## 1.4 Data Storage and Communication

# 2 Discussion per Class

## 2.1 Board

Board maakt gebruik van de classes: Color en Field.

Zoals de naam al doet vermoeden, vervult de Board class de rol van het speelbord van Connect Four 3D. Board is dus verantwoordelijk voor de representatie van het speelbord. Board zorgt ervoor dat er een bord wordt gecreëerd met een grootte van 4 bij 4 bij 4. In eerste instantie zijn alle velden op het bord leeg. Alle velden op het bord hebben een x-, een y- en een z-coördinaat.

Naast de representatie van het speelbord is Board ook verantwoordelijk voor het controleren van een zet (oftewel: controleren of een zet een geldige zet is).

## 2.2 Color

De class Color representeert een steen in het Connect Four 3D spel. Een steen kan drie verschillende waarden hebben, namelijk: RED, YEL en EMP. Als een veld op het bord leeg is, krijgt deze de kleur EMP. Omdat er sprake is van een vooraf gedefinieerde set van kleuren, is Color een enum.

Color maakt geen gebruik van andere classes.

## 2.3 Field

De class Field representeert een veld op het Connect Four 3D bord. Een field heeft een bepaalde x-coördinaat, y-coördinaat, x-coördinaat en een kleur. Field is een relatief simpele class waarin het mogelijk is om de coördinaten en/of de kleur van een bepaald vakje op te vragen. Verder heeft Field nog een set functie waarmee het mogelijk is om de kleur van een leeg vakje aan te passen naar de gewenste kleur.

De class Field maakt gebruik van de class Color.

## 2.4 Player

In de class Player wordt er een nieuwe speler aangemaakt. Deze speler krijgt in de constructor een naam en een lege hand van Color objecten mee.

Met de methodes getName en getColor is het voor elke speler is het mogelijk om de naam en kleur op te vragen. Met de methodes setName en setColor is het mogelijk om de naam en kleur aan te passen. Met de methode showHand kan het aantal stenen dat de speler nog in zijn hand heeft opgevraagd worden. Met de reset methode krijgt een speler weer 32 nieuwe stenen van één bepaalde kleur. In initHand worden er 32 stenen voor de speler gecreëerd en wordt er aan deze stenen ook één kleur meegegeven. In clearHand wordt er weer een lege hand van Color objecten voor de speler gemaakt. Met getGame is het mogelijk om de huidige game op te vragen.

In makeMove wordt de speler door de methode determineMove eerst gevraagd in welke kolom en rij hij een steen wil plaatsen. Vervolgens vraagt makeMove de coördinaten van de gekozen locatie en de kleur van de speler op. Met deze informatie kan makeMove met behulp van setField de steen daadwerkelijk op het bord plaatsen.

De class Player maakt gebruik van de classes Color, Game en Field.

## 2.5 Humanplayer

De class Humanplayer is een uitbreiding van Player. Een human player is een speler die zelf bepaalt waar zijn stenen op het bord geplaatst moeten worden.

Als uitbreiding op Player heeft Humanplayer een methode determineMove waarin de zet van de speler wordt bepaald. Om de gewenste kolom te bepalen, wordt eerst de methode askColumn aangeroepen. Deze methode vraagt door middel van een print line aan de speler in welke kolom de steen geplaatst moet worden. De input van de speler wordt uitgelezen en vervolgens doorgegeven aan determineMove. Om de gewenste rij te bepalen, wordt de methode askRow aangeroepen. Deze methode vraagt ook door middel van een print line aan de speler in welke rij de steen geplaatst moet worden. Aan de hand van de ingevoerde rij en kolom, kan de y-positie van de steen bepaald (de steen valt immers altijd zo ver mogelijk naar beneden).

Humanplayer maakt gebruik van de classes Board, Field en Color.

## 2.6 Computerplayer

Net zoals Humanplayer is ook Computerplayer een uitbreiding van Player. Bij een computer player plaatst de computer “zelf” een steen op het bord.

In de determineMove wordt er met een for loop voor de x-, y- en z-positie gezocht naar een leeg veld op het bord. Zodra er een leeg veld is gevonden, wordt de steen van de computer speler op dat veld geplaatst.

Met de huidige implementatie van determineMove is de kans dat een computer speler een game wint niet erg groot. We zijn er helaas niet in geslaagd om een slimmere strategie te implementeren binnen het opgegeven tijdsbestek.

## 2.7 Game

De class Game representeert een game binnen het Connect Four 3D spel.

In de constructor van Game wordt er een nieuwe game met twee spelers gecreëerd. Aan deze twee spelers wordt een kleur meegegeven. Verder wordt er in de constructor een nieuwe view gemaakt.

De methode getView levert de huidige view op en setView past de huidige view aan naar de gegeven parameter. De methode getCurrentPlayer geeft de huidige speler en changePlayer past de huidige speler aan naar de volgende speler. De methode firstPlayer bepaalt willekeurig welke speler het spel mag beginnen. De methode isWinner kijkt of een meegegeven kleur ergens vier op een rij op het bord heeft. Dit doe hij door getXdiag, getDiagDiag, getZdiag, getCol, getZRow, getXRow op het meegegeven bord aan te roepen. Met de methode hasWinner wordt er bepaald of er een winnaar is op het huidige bord. Een game kan gereset worden met de methode reset. Met de methode start kan er een game gestart worden. Daarnaast vraagt start de speler of de regels van het spel geprint moet worden en of de speler gebruik wil maken van de hint functie tijdens de game. Als de spelregels geprint moeten wordt, wordt printRules aangeroepen. Deze methode bevat en print alle regels van het spel. Als de speler gebruik wil maken van de hint functie, wordt de methode getHint aangeroepen. Deze methode geeft aan wat de beste positie is voor een speler om zijn/haar steen te plaatsen.

Met readBoolean kan er een vraag aan een speler worden gesteld. Een vraag kan beantwoord worden met “y”, “yes”, “n” en “no”. readBoolean checkt of de vraagt met “y” of “yes” beantwoord is en als dit het geval is, wordt er actie ondernomen (denk bijvoorbeeld aan het printen van de spelregels). Met play kan er een game op het gegeven bord gespeeld worden. De play methode zorgt ervoor dat een speler een zet kan doen en dat er van speler wordt gewisseld als een speler een zet heeft gedaan. Daarnaast vraagt play bij elke zet of de speler een hint nodig heeft. Deze vraag wordt vervolgens weer door readBoolean uitgelezen.

De methode gameOver controleert of er al een winnaar is en controleert ook of het bord vol is. Als één van beide gevallen waar is, is de game afgelopen. Als een game is afgelopen, print printResult de winnaar met zijn/haar naam en kleur. Als er geen winnaar is, omdat het bord vol is, wordt dit ook aangegeven. Om te bepalen wie de winnaar is, roept printResult de methode winner aan. Deze methode levert de speler die gewonnen heeft op.

De methode update print de huidige situatie van een game door het bord met daarop de geplaatste stenen te printen. Daarnaast wordt ook de huidige speler met zijn/haar naam en kleur geprint.

Game maakt gebruik van de classes Player, Board, TUIView, Color, Field en Humanplayer.

## 2.8 ThreeDRow

In de main methode van ThreeDRow worden er twee nieuwe spelers aangemaakt. Om te bepalen of een speler een Computer- of een Humanplayer is, moeten er twee argumenten aan deze class meegegeven worden. Voor een Humanplayer moet het argument “-H” meegegeven worden en voor een Computerplayer het argument “-C”. Voor een game met twee Humanplayers wordt er bij de Program arguments dus “-H -H” ingevuld. Verder wordt er in de main een nieuwe game met de twee spelers aangemaakt en wordt de game gestart.

ThreeDRow maakt gebruik van de classes Game, Player, Color, Humanplayer en Computerplayer.

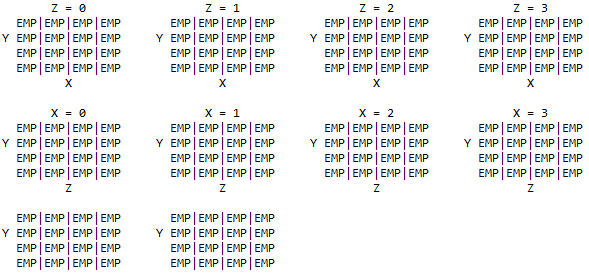
## 2.9 Protocol

In het protocol staat gedefinieerd hoe de communicatie tussen de client en de server moet verlopen. Dit maakt het voor alle spelers die gebruik maken van hetzelfde protocol mogelijk om Connect Four 3D via een server te spelen.

## 2.10 TUIView

De class TUIView is verantwoordelijk voor de weergave van het bord. TUIView dit doet in de vorm van een TUI.

Het bord kan opgedeeld worden in 10 verticale vlakken: 4 vlakken in de z-richting, 4 vlakken in de x-richting en 2 diagonale vlakken. De TUI ziet er in de begintoestand als volgt uit:



## 2.11 Client

In de main methode van Client wordt een client applicatie gestart. Om een client applicatie te kunnen starten, moet de speler een adres en een poort invullen bij de Program arguments van de class Client. Als zowel het adres als de poort geldig zijn, worden er een join- en gamerequest verzonden. Vervolgens wordt de client gestart.

## 2.12 ClientHandler

Voor alle spelers die verbonden zijn met de server wordt er een ClientHander aangemaakt. ClientHandler zorgt ervoor dat alle commando’s die door de client worden verstuurd, worden geïnterpreteerd en uitgevoerd. In de methode handleInput wordt bepaald welk commando de client heeft verstuurd. Vervolgens wordt , afhankelijk van het commando, één van de methodes handleJoinReq, handleJoinGameReq, handleSetMove aangeroepen om het commando uit te laten voeren.

## 2.13 GameHandler

## 2.14 Server

De class Server representeert de centrale server voor Connect Four 3D. Server is verantwoordelijk voor het tot stand brengen van een connectie tussen twee spelers (clients) en speelt dus een grote rol in de synchronisatie van de spellen die gespeeld worden.

# 3 Test report

## 3.1 Unit Testing

### 3.1.1 BoardTest

Om de methodes van board goed te kunnen testen, wordt er in de setup van de Board Test een leeg bord gecreëerd.

In de eerste test wordt er gekeken of de isField methode werkt door isField de volgende coördinaten mee te geven: (-1, -1, -1), (0, 0, 0), (3, 3, 3) en (4, 4, 4).

In de tweede test wordt er gekeken of de getField methode werkt door een bepaald veld met setField een rode kleur mee te geven en daarna te controleren of getField op dit veld inderdaad een rode kleur teruggeeft.

In de derde test wordt de setField methode getest door te controleren of deze methode wel maar aan één vakje een bepaalde kleur meegeeft. Een vakje krijgt met de setField methode een rode kleur. Vervolgens wordt er met de getField methode op verschillende coördinaten gecontroleerd of deze vakjes niet ook toevallig een rode kleur hebben gekregen.

In de vierde test wordt er gekeken of de reset methode goed werkt door eerst de reset methode aan te roepen op een bord en vervolgens te controleren of het eerste vakje op het bord de kleur EMP heeft.

In de vijfde test wordt er gekeken of de deepCopy methode goed werkt. Eerst wordt er aan het (0, 0, 0) vakje een rode kleur meegegeven. Vervolgens wordt er een deepCopy van dit bord gemaakt en wordt er gecontroleerd of de inhoud van de deepCopy overeen komt met de inhoud van het bord. Daarna wordt er ook nog gekeken of het bord en de deepCopy als twee aparte borden gebruikt kunnen worden. Dit wordt gecontroleerd door het vakje (0, 0, 0) op de deepCopy een gele kleur mee te geven en vervolgens te kijken of het normale bord nog steeds een rode steen heeft en de deepCopy een gele steen.

In de zesde test wordt de methode isEmpty getest. Het (0, 0, 0) vakje krijgt een rode kleur en vervolgens wordt er gekeken of isEmpty false oplevert als de methode wordt aangeroepen op (0, 0, 0).

Bij testHasRowX, testHasRowZ, testHasColumn, testGetDiagDiag, testGetXDiag1, testGetXDiag2, testGetZDiag1 en testGetZDiag2 worden er met behulp van setField steeds 4 stenen van één kleur op een rij geplaatst. Vervolgens kunnen de bovenstaande methodes aangeroepen worden op het 4 op een rij bord en wordt er gekeken of de methodes inderdaad true opleveren.

In testValidMove wordt er eerst steeds een steen geplaatst op een leeg vakje op het bord. Er wordt dan gecontroleerd of validMove dan true oplevert. Vervolgens wordt dezelfde steen met setField nogmaals op hetzelfde vakje geplaatst en wordt er gecontroleerd of validMove false oplevert.

Board heeft een test coverage van 97,3 %.

### 3.1.2 FieldTest

In de setUp van FieldTest wordt er een nieuw veld geïnitieerd met (1, 1, 1) als coördinaten en kleur EMP.

In de test wordt er eerst gekeken of getColor op het veld (1, 1, 1) ook de kleur EMP oplevert. Vervolgens geeft setColor het veld een gele kleur mee. Er wordt dan gekeken of getColor op het veld (1, 1, 1) nu ook de kleur geel heeft. Daarna worden de methodes getX, getY en getZ getest door te kijken of zij allemaal 1 opleveren. Als een vakje een rode of gele kleur heeft, mag het niet mogelijk zijn om de kleur van het vakje te veranderen. Dit wordt getest door het veld (1, 1, 1) een rode kleur mee te geven en vervolgens te controleren of getColor niet gelijk is aan rood.

Field heeft een test coverage van 100 %.

### 3.1.3 GameTest

In de setUp van de GameTest worden er eerst twee nieuwe HumanPlayers geïnitieerd. Vervolgens wordt er met deze twee spelers een nieuwe game gemaakt. Verder wordt er ook een nieuw bord geïnitieerd.

In testIsWinner wordt er eerst een kleur toegewezen aan de beide spelers. Vervolgens wordt er met setField een rode rij in de x-richting gecreëerd. Tot slot wordt er gecontroleerd of isWinner true oplevert voor de rode kleur.

Test uitbreiden!

### 3.1.4 PlayerTest

In de setUp van PlayerTest worden er twee nieuwe HumanPlayers geïnitieerd. Verder wordt er ook een nieuw bord gemaakt.

In testGetName wordt er gekeken of de getName functie dezelfde namen oplevert als de namen die aan de HumanPlayers zijn meegegeven in de setUp van de test.

In testGetSetColor wordt de kleur rood aan speler 1 meegegeven en de kleur geel aan speler 2. Vervolgens wordt er gekeken of de getColor methode de kleuren oplevert die in de setColor methode zijn meegegeven.

In testSetName worden de namen van de spelers, die in de setup zijn meegegeven, met de setName methode verandert. Vervolgens wordt er gekeken of de getName methode nu ook de nieuwe namen oplevert.

In testShowHand en testInitHand wordt er voor één speler eerst een initiële hand met 32 stenen gecreëerd. Vervolgens wordt er gekeken of de showHand methode ook 32 stenen teruggeeft. Vervolgens wilden we met testShowHand testen of showHand 31 op zou leveren als de speler een move had gedaan, maar dit is niet gelukt.

In testClearHand wordt er net zoals bij testShowHand en testInithand eerst een initiële hand met 32 gecreëerd. Vervolgens wordt er gekeken of de showHand methode ook 32 stenen teruggeeft. Daarna wordt de clearHand methode aangeroepen en wordt er gekeken of de showHand methode nu 0 stenen oplevert.

In testMakeMove krijgt speler 1 de kleur rood. Vervolgens wordt de speler via de makeMove methode gevraagd om de gewenste rij en kolom voor de move in te vullen. Daarna wordt er met de getField methode gecontroleerd of het door de speler geselecteerde veld ook de rode kleur heeft.

Player heeft een test coverage van 83,2 %.

## 3.2 System testing

# 4 Metrics report

# 5 Reflection on Planning

Doordat we het programmeerproject vorig jaar ook hebben gedaan, is het maken van een planning ons goed af gegaan. We wisten inmiddels ongeveer hoe veel tijd we nodig zouden hebben voor alle onderdelen van het project en dit maakte het maken van een planning een stuk makkelijker.

## 5.1 Planning

|  |  |  |
| --- | --- | --- |
| Datum | Tijd | To do |
| 12 december | - | Start programmeerproject |
| 17 januari | 12:00 - 17:30 | Client/Server |
| 23 januari | 12:30 – 17:30 | Client/Server |
| 24 januari | 9:30 - 12:30 | Client/Server |
| 25 januari | 10:00 - 17:30 | Client/Server |
| 26 januari | 9:30 - 12:30 | Client/Server |
| 27 januari | 12:00 - 16:00 | Verslag + Tactiek ComputerPlayer |
| 30 januari | 9:30 - 17:30 | Implementeren ComputerPlayer + Verslag |
| 31 januari | 9:30 - 17:30 | Puntjes op de i + Verslag |
| 1 februari | 9:30 - 17:30 | Puntjes op de i + Verslag |
| 1 februari | 23:59 DEADLINE |  |