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

//TODO: Michael client fixen.

### Observer

Bij de server wordt het board geobserveerd door de ServerGUI. Door de directe connectie tussen Server en de ServerGUI was het niet nodig om server te observeren aangezien de server het direct aan de view kon doorgeven. Maar het board werd niet bijgehouden door de server, alleen door de ClientHandlers. Dus door het board te observeren kon de ServerGUI makkelijk weergeven of er een board aangepast was,

### Model-View-Controller

Bij de server is het model de Classen board en server, board bevat alle logica voor de Connect4 game en server bevat de logica om invites, ClientHandlers en het leaderboard bij te houden. De controllers zijn de ClientHandlers en deels de server. De server zorgt ervoor dat ClientHandlers worden aangemaakt. De ClientHandlers vangen de commando’s van de clients op en voeren de bijbehorende methodes uit. Het view is de ServerGUI die de berichten weergeeft.

## 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:

* HELP: Stuurt door aan de view welke commando’s er op dit moment gebruikt kunnen worden, afhankelijk van of de Client ingame is.
* MOVE: Checkt of er om een move gevraagd is. Zo ja, dan stuurt hij de move door naar de Server, zo nee, dan geeft hij een foutmelding.
* INVITE: Voegt de invite toe aan de lijst met invites. Als er een custom board size mee is gegeven, dan wordt deze ook opgeslagen.
* ACCEPT: Stuurt de Server een pakket waarin staat dat de invite van de speler geaccepteerd is.
* DECLINE: Stuurt de Server een pakket waarin staat dat de invite van de speler gedeclined is.
* HINT: Als de Client in-game is, regel dat er een hint wordt doorgegeven aan de Client, anders, geef een foutmelding.
* DIFFICULTY: Verandert de moeilijkheid van de ComputerPlayer (het aantal zetten dat deze vooruit denkt).

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

ClientTUI voor het vragen om input en om berichten aan door te geven, Board om het bord bij te houden en ComputerPlayer voor het maken van hints, en eventueel zetten te doen.

### Voorzorgsmaatregelen om de precondities in the server te vervullen

Waar nuttig zijn checks geïmplementeerd, zoals het checken van de lengte van de commando’s die je stuurt.

## ClientHandler.java

### Rol van de klasse in het systeem

ClientHandler.java regelt het opvangen van alle commando’s die de client stuurt en de bijbehorende methodes uitvoeren. Ook houdt de ClientHandler de game bij samen met de ClientHandler van de tegenstander.

### De verantwoordelijkheden van deze klasse

Deze klasse is verantwoordelijk voor het aanmelden van een client met behulp van het connect command. Dus deze klasse is ook verantwoordelijk voor het controleren van de naam en de features van de client en het opvangen van alle andere commando’s van de client en het bijbehorende antwoord geven. Deze klasse is ook verantwoordelijk voor het afmelden van de client als de verbinding verbroken wordt, voor het maken en verwijderen van invites (maar niet voor het bijhouden van de invites) en voor het bijhouden van het leaderboard. Verder is deze klasse nog verantwoordelijk voor het starten van een game, het bijhouden van het bord, controleren of de moves wel kunnen en de beurt van de volgende speler doorgeven of aangeven dat er iemand heeft gewonnen.

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

Deze klasse gebruikt server voor het uitvoeren van de meeste commando’s en het communiceren met andere ClientHandlers. Verder wordt LeaderboardPair gebruikt om een gesorteerd leaderboard bij te houden.

### Speciale gevallen in deze klasse’s contract

### Voorzorgsmaatregelen om de precondities in the server te vervullen

## ClientTUI.java

### Rol van de klasse in het systeem

ClientTUI is de TUI (Textual User Interface), die de gebruiker te zien krijgt zodra hij het programma opstart. Deze klasse regelt het vragen van de naam, IP-adres en poort. Zodra deze bekend zijn, en er een succesvolle verbinding met de server tot stand is gebracht, wordt geluisterd naar commando’s van System.in. Daarnaast laat het berichten aan de gebruiker zien.

### De verantwoordelijkheden van deze klasse

Deze klasse is verantwoordelijk voor het doorsturen van ingevoerde commando’s naar Client.java, en het tonen van berichten die door Client zijn verstuurd.

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

Client voor het ontvangen van commando’s en het doorsturen van berichten die moeten worden laten zien in de TUI.

### Voorzorgsmaatregelen om de precondities in the server te vervullen

Geen.

## ClientView.java

### Rol van de klasse in het systeem

ClientView is een interface. Deze wordt door ClientTUI geïmplementeerd. Het bevat een aantal methoden die een UI sowieso moet hebben. Mochten we kiezen om een GUI te maken, dan kan die ook deze interface implementeren.

## Features.java

### Rol van de klasse in het systeem

Bevat de constanten voor de Features die een Client en Server kunnen hebben, die vastgelegd zijn in het protocol.

## LeaderboardPair.java

//TODO: Laat Sven dit fixen.

### Rol van de klasse in het systeem

### De verantwoordelijkheden van deze klasse

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

### Speciale gevallen in deze klasse’s contract

### Voorzorgsmaatregelen om de precondities in the server te vervullen

## MessageUI.java

### Rol van de klasse in het systeem

Deze interface wordt gebruikt als een globale interface voor UI’s. Deze klasse wordt geextend door ClientView en geïmplementeerd door ServerGUI.

## Server.java

//TODO: Laat Sven dit fixen.

### Rol van de klasse in het systeem

### De verantwoordelijkheden van deze klasse

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

### Speciale gevallen in deze klasse’s contract

### Voorzorgsmaatregelen om de precondities in the server te vervullen

## ServerGUI.java

### Rol van de klasse in het systeem

Geeft het IP van de server weer, en laat de gebruiker een poort kiezen waarop de Server gestart moet worden. Laat na een succesvolle start alles zien wat er op de server gebeurt. Mocht de poort al in gebruik zijn door een andere applicatie, dan wordt er een foutmelding gegeven.

### De verantwoordelijkheden van deze klasse

Een server starten op de juiste poort en alle berichten die doorgegeven worden aan deze klasse laten zien in het berichtvak.

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

Server.java

### Voorzorgsmaatregelen om de precondities in the server te vervullen

Als de ingevoerde poort geen integer is, wordt er een foutmelding weergegeven.

## Board.java

### Rol van de klasse in het systeem

Het regelen van een Board object. Dit houdt onder andere in: moves maken en kijken of een speler gewonnen heeft.

### De verantwoordelijkheden van deze klasse

De moves die doorgegeven worden maken op het bord en op een correcte manier kijken of een speler gewonnen heeft.

## BoardTest.java

### Rol van de klasse in het systeem

Een testklasse voor Board.java. Deze klasse test iedere methode van Board waarvoor dit zinvol is.

### De verantwoordelijkheden van deze klasse

Board.java feilloos testen.

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

Board.java

### Voorzorgsmaatregelen om de precondities in the server te vervullen

Er worden geen onmogelijke bordsituaties getest.

## ComputerPlayer.java

### Rol van de klasse in het systeem

Beheert een ComputerPlayer met een Strategy. Wordt vooral gebruikt vanwege zijn determineMove methode om een move te maken of om een hint te vragen.

### De verantwoordelijkheden van deze klasse

Een valide move returnen volgens zijn Strategy.

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

\*Strategy.java.

## Disc.java

### Rol van de klasse in het systeem

Dit is een enum om de verschillende Discs bij te houden. Een Disc kan YELLOW, RED of EMPTY zijn.

## MinMaxStrategy.java

//TODO: Laat Sven dit fixen.

### Rol van de klasse in het systeem

Dit is onze implementatie van de MinMaxStrategy voor een ComputerPlayer. Deze Strategy denkt enkele zetten vooruit.

### De verantwoordelijkheden van deze klasse

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

### Speciale gevallen in deze klasse’s contract

### Voorzorgsmaatregelen om de precondities in the server te vervullen

## NaiveStrategy.java

### Rol van de klasse in het systeem

Deze klasse is voor een ComputerPlayer, die iedere keer een compleet random move op een Board doet.

### De verantwoordelijkheden van deze klasse

Een random valide move op het Board returnen.

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

Board.java.

## SmartStrategy.java

### Rol van de klasse in het systeem

Deze klasse is voor een ComputerPlayer, die iedere keer een zet vooruit denkt, en dan een move op een Board doet.

### De verantwoordelijkheden van deze klasse

Maak een move waarbij een move wordt gedaan op een Board waarbij 1 zet vooruit is gedacht.

### De andere klassen die door deze klasse gebruikt worden om zijn verantwoordelijkheden te bereiken

Board.java.

## Strategy.java

### Rol van de klasse in het systeem

Een interface voor de andere Strategy. Alle andere strategies moeten ten minste een getName en een determineMove methode hebben.

# Test report

Hieronder vindt u per klasse de testgegevens over die klasse. Als een kopje niet van toepassing was op die klasse, hebben we hem weggelaten.

## ClientHandler.java

//TODO: Laat Sven dit fixen.

### Hoe is deze klasse getest?

### Welke test techniek is toegepast?

### Welke testprogramma’s zijn ontwikkeld?

### Wat waren de verwachte resultaten en de echte resultaten?

### Hoeveel procent van iedere methode wordt gedekt door tests?

## ClientTUI.java

//TODO: Discussiëren hoe we deze klasse gaan testen.

### Hoe is deze klasse getest?

### Welke test techniek is toegepast?

### Welke testprogramma’s zijn ontwikkeld?

### Wat waren de verwachte resultaten en de echte resultaten?

### Hoeveel procent van iedere methode wordt gedekt door tests?

## ClientView.java

### Hoe is deze klasse getest?

Het is een interface. Deze klasse is niet getest.

## Client.java

//TODO: Discussiëren hoe we deze klasse gaan testen.

### Hoe is deze klasse getest?

### Welke test techniek is toegepast?

### Welke testprogramma’s zijn ontwikkeld?

### Wat waren de verwachte resultaten en de echte resultaten?

### Hoeveel procent van iedere methode wordt gedekt door tests?

## Features.java

### Hoe is deze klasse getest?

Deze klasse bevat alleen constanten. Deze klasse is niet getest.

## LeaderboardPair.java

//TODO: Laat Sven dit fixen.

### Hoe is deze klasse getest?

### Welke test techniek is toegepast?

### Welke testprogramma’s zijn ontwikkeld?

### Wat waren de verwachte resultaten en de echte resultaten?

### Hoeveel procent van iedere methode wordt gedekt door tests?

## MessageUI.java

### Hoe is deze klasse getest?

Het is een interface. Deze klasse is niet getest.

## Server.java

//TODO: Laat Sven dit fixen.

### Hoe is deze klasse getest?

### Welke test techniek is toegepast?

### Welke testprogramma’s zijn ontwikkeld?

### Wat waren de verwachte resultaten en de echte resultaten?

### Hoeveel procent van iedere methode wordt gedekt door tests?

## ServerGUI.java

//TODO: Laat Sven dit fixen.

### Hoe is deze klasse getest?

### Welke test techniek is toegepast?

### Welke testprogramma’s zijn ontwikkeld?

### Wat waren de verwachte resultaten en de echte resultaten?

### Hoeveel procent van iedere methode wordt gedekt door tests?

## Board.java

### Hoe is deze klasse getest?

Board.java is getest in de klasse BoardTest.java. Voor vrijwel iedere methode is een zinvolle test ontwikkeld. Voor enkele methoden was het niet zinvol om een test te schrijven, bijvoorbeeld omdat deze niets anders deden, dan te checken of 1 van de 2 calls naar 2 andere methoden true waren. Bij deze methoden hebben we in BoardTest.java een stub gemaakt, met commentaar erin. In dat commentaar staat de reden, dat we die methoden niet getest hebben.

### Welke test techniek is toegepast?

De tests zijn ontwikkeld met behulp van JUnit. De documentatie voor de tests staat bij de code in de vorm van Javadoc.

### Welke testprogramma’s zijn ontwikkeld?

Voor deze klasse zijn geen aparte testprogramma’s ontwikkeld.

### Wat waren de verwachte resultaten en de echte resultaten?

Dit staat in het commentaar bij de desbetreffende code. Na enkele revisies van Board.java kwamen de verwachte resultaten overeen met de echte resultaten.

### Hoeveel procent van iedere methode wordt gedekt door tests?

Emma gaf ons een globaal percentage van 85,1%. Alle methoden zijn óf voor 100% getest, of voor 0%. De methoden, die voor 0% getest zijn, zijn: gameOver(), hasWinner(), isWinner(Disc) en toString().

## BoardTest.java

### Hoe is deze klasse getest?

Deze klasse is getest door te kijken hoeveel van Board hij test.

### Welke test techniek is toegepast?

Hij is getest aan de hand van de coverage die de EMMA plugin gaf.

### Welke testprogramma’s zijn ontwikkeld?

EMMA, maar niet door ons.

### Wat waren de verwachte resultaten en de echte resultaten?

We gingen voor een percentage boven de 80, en we kregen 85,1%.

### Hoeveel procent van iedere methode wordt gedekt door tests?

EMMA dekt alle methoden van deze klasse.

## ComputerPlayer.java

//TODO: Discussiëren hoe we deze klasse gaan testen.

### Hoe is deze klasse getest?

### Welke test techniek is toegepast?

### Welke testprogramma’s zijn ontwikkeld?

### Wat waren de verwachte resultaten en de echte resultaten?

### Hoeveel procent van iedere methode wordt gedekt door tests?

## Disc.java

### Hoe is deze klasse getest?

Het is een enumklasse. Deze klasse is tijdens het ontwikkelen meer dan voldoende getest, tijdens het spelen van testgames. Hij is lang niet meer aangepast.

### Welke test techniek is toegepast?

Visuele inspectie van TUI.

### Welke testprogramma’s zijn ontwikkeld?

Geen.

### Wat waren de verwachte resultaten en de echte resultaten?

Verwacht: Laat R zien voor alle rode Discs, Y voor alle gele Discs en niks voor empty Discs. Dit gebeurde ook.

### Hoeveel procent van iedere methode wordt gedekt door tests?

100%.

## NaiveStrategy.java

### Hoe is deze klasse getest?

Spelletjes spelen tegen deze Strategy.

### Welke test techniek is toegepast?

Visuele inspectie van de TUI.

### Welke testprogramma’s zijn ontwikkeld?

Geen.

### Wat waren de verwachte resultaten en de echte resultaten?

Dat hij random zetten zou doen, en dat deed hij ook.

### Hoeveel procent van iedere methode wordt gedekt door tests?

De getName methode is niet getest, maar deze wordt ook niet gebruikt. determineMove is geheel getest.

## SmartStrategy.java

### Hoe is deze klasse getest?

Spelletjes spelen tegen deze Strategy.

### Welke test techniek is toegepast?

Visuele inspectie van de TUI.

### Welke testprogramma’s zijn ontwikkeld?

Geen.

### Wat waren de verwachte resultaten en de echte resultaten?

Dat hij redelijk slimme zetten zou doen, en dat deed hij ook.

### Hoeveel procent van iedere methode wordt gedekt door tests?

De getName methode is niet getest, maar deze wordt ook niet gebruikt. determineMove is geheel getest.

## Strategy.java

### Hoe is deze klasse getest?

Het is een interface. Deze klasse is niet getest.

# Metrics report

# Reflectie op de planning

## Hoe was je planning beinvloed door jouw ervaringen met planning en tijdschrijven in week 4 van de module?

Op geen enkele manier.

## In hoeverre kwam je planning overeen met de werkelijkheid?

Het project liep zo voorspoedig, dat we telkens stukken voorliepen op de planning. Hierdoor werd de planning niet vaak tevoorschijn gehaald, maar als we dat deden, dan zagen we dat we alweer een paar dagen voor op schema liepen.

## Welke tegenmaatregelen heb je genomen om afwijking van de originele planning te compenseren? Wat was de impact hiervan op de kwaliteit van ons project?

We hebben extra features toegevoegd, hierdoor ging de kwaliteit van ons project omhoog.

## Wat heb je geleerd van deze ervaring voor je volgende (project)planning?

Als je voorloopt op de planning, heb je geen stress, en komt alles op tijd af. Ook heb je dan nog voldoende tijd om extra features toe te voegen.

## Twee do’s en don’t’s voor studenten die volgend jaar dit project gaan doen

Do: Begin op tijd. Wij zijn 2 weken voor de deadline serieus begonnen, en hebben iedere weekdag van 10 tot 5 besteed aan het werken aan het project. Ervoor hadden we al een klein opzetje met klassen die mogelijk nuttig konden zijn. Verder: Spreek een plek af, waar je iedere dag zit. Wij wonen 30 kilometer van de universiteit af in hetzelfde dorp. De ene dag gaan we naar die persoon’s huis, de andere dag naar de andere persoon’s huis. Dit scheelt reistijd en is duidelijk voor ons allebei.

Don’t: Niet op tijd beginnen. Als je nog niet iets hebt 1 week voor de deadline: good luck. Verder: Niet tot laat ’s avonds doorwerken, en dan uitslapen, en dan weer tot laat doorwerken. Niet goed voor de motivatie en je bent meer tijd kwijt met slapen dan je denkt.