Programmeer Project

Josje van 't Padje

S1423037

Inhoudsopgave

Inleiding

In de afgelopen module hebben we, vooral de laatste paar weken, gewerkt aan een project. In dit project was het de bedoeling dat we een bordspel zouden maken wat we over een server gespeeld kan worden. Het bordspel wat wij moesten gaan implementeren stond al vast, ROLIT.

ROLIT is een spel wat gespeeld kan worden door 2 tot 4 personen. Deze personen hebben allemaal hun eigen kleur, rood, geel, groen of blauw. Tijdens het spelen van het spel legt elke speler om de beurt een balletje in een van de vakjes van het bord met zijn eigen kleur omhoog. Als er dan een ander balletje met de zelfde kleur in een van de acht richtingen om het zojuitst gelegde balletje ligt moeten alle balletjes tussen het net gelegde balletje en het andere balletje met de zelfde kleur zo gedraaid worden dat de kleur van die speler boven ligt. Dit heet slaan. De balletjes die je legt moeten aanliggend zijn aan de balletjes die al liggen en als je kan slaan, moet je slaan.

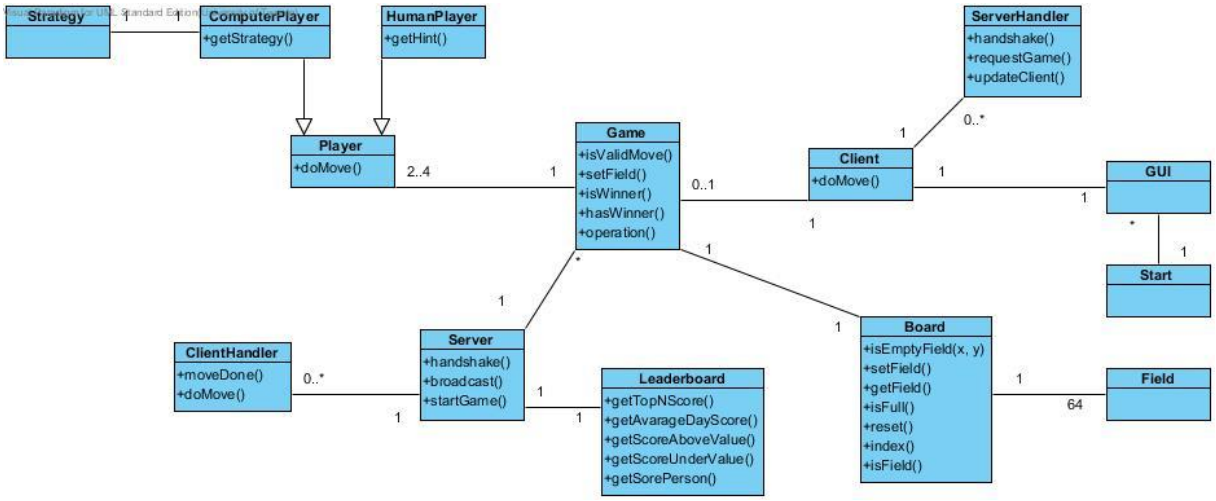
Het spel begint met de beginpositie, dat zijn vier balletjes in de vier kleuren rond het midden van het bord. Rood is het eerst aan slag. Daarna geen, dan groen en dan blauw. Als er met 3 mensen gespeeld wordt veranderd er niets aan de begin positie, alleen valt speelt blauw gewoon niet mee, die zal dus al snel van het spelbord verdwijnen. Met twee mensen spelen alleen rood en groen tegen elkaar. Het uiteindelijke doel van het spel is om zoveel mogelijk balletjes met jou kleur boven op het spelbord te hebben liggen.

Het spel moet instaat zijn deze regels na te leven en de vakjes van kleur te laten veranderen als er een slag gezet is. Verder is het de bedoeling dat het spel door het aangegeven aantal mensen gespeeld moet kunnen worden op verschillende PC's en dat je ook een AI voor jou moet kunnen laten spelen.

In dit verslag beschrijf ik bij 'Discussion of the Overal Design' eerst wat ik tot nu toe heb en daarna wat er nog bij had gemoeten. De klasses zoals ze zijn en soms zoals ze zouden moeten worden als het spel af was gekomen.

Discussion of the overall Design:

Class Diagrams



Verplichte requirements

Een vereiste is dat de client een GUI moet hebben. Deze wordt gemaakt door verscheidene klassen. Zo is er een klasse voor de client waar de serveradres en het poortnummer kan worden ingevuld en waar kan worden geselecteerd met hoeveel spelers de client wil spelen. Daarnaast is er een GUI voor het werkelijke spel zelf. Hier kan de client het spel Rolit spelen door op de verschillende vakjes te drukken die al dan niet van kleur veranderen. Ook is er voor de client een mogelijkheid om een hint op te vragen en een nieuw spel te starten.

Er wordt gevraagd om de client support te geven voor zowel menselijke als computer spelers. Dit wordt waargemaakt doordat er in het startscherm kan worden gekozen tussen een menselijke of een computerspeler. Bij een computerspeler kan dan een strategie worden gekozen die is geimplementeerd in de klasse NaiveStrategy. Als er voor een menselijke speler wordt gekozen, kan er worden gespeeld met een muis en wordt de klasse HumanPlayer aangeroepen. En als er een computer wordt gekozen, wordt de klasse ComputerPlayer aangeroepen. Zowel HumanPlayer als ComputerPlayer implementeren de klasse Player. Deze klasse is een interface die voor alle spelers gelijk is. Helaas werkt mijn rolit nog niet met Players dus zul je nogsteeds zelf moeten spelen ookal geef je aan dat je met een computerplayer wilt spelen.

De speler kan in het Start scherm wel zelf een ip adress en een poortnummer kiezen. Dit is dus een ClientGUIAls hierna op de connect knop geklikt wordt wordt er verbinding gemaakt met de server. De client stuur “hello [naam] [opties] [versie]” naar de ClientHandler, die stuurt dan “hello [opties] [versie]” terug en de ClientHandler laat nog een bericht door de server broadcasten dat de client online is. En doet dit weer wanneer de client weer offline gaat.

Het spel moet ook een mogelijkheid om een nieuw spel te starten als het spel is afgelopen. Dit kan door een knop binnen de spelGUI die actief wordt als het bord vol is en dus als het spel is afgelopen. Deze knop roept de RolitController aan. Deze klasse reset dan de game via de reset methode die in de Game klasse zit.

Andere vereiste, die bijvoorbeeld aan de server gesteld zijn heb ik helaas niet afgekregen, mijn server is in staat te zien en te melden of een client online of offline is en hello te zeggen tegen de client als de client hello gezegt heeft.

Optionele requirements

Als eerste requirement is er de hint request requirement. Hierin wordt gevraagd naar een mogelijkheid om een hint op te vragen. Dit wordt waargemaakt door een knop binnen de GUI die men tijdens het spelen kan gebruiken om tijdens het spelen alle mogelijke zetten tijdens de huidige beurt een andere kleur te geven. Deze mogelijke zetten zijn dus niet altijd zichtbaar, enkel als er om wordt gevraagd. Deze knop roept de RolitController aan die vervolgens de lijst met mogelijke zetten opvraagd van de Game klasse. Daarna worden een voor een alle mogelijke zetten op de GUI een andere kleur gegeven.

Verder heb ik een leaderboard, deze is nog niet aan het spel gekoppeld maar is wel in staat alle gegvens op te slaan en de gewenste informatie te tonen als hier om gevraagd wordt.

De observer en MVC

Als observer is gekozen voor de GUI, die als observable de Game klasse heeft. De GUI is dan bij ons de view en Game is de model. Als controller zit de RolitController klasse tussen. Deze controller zorgt ervoor dat er kan worden gecommuniceerd tussen de GUI en de Game. De Game notified de observers als er wat moet worden veranderd binnen de GUI. Binnen de GUI klasse wordt de GUI dan ook als oberver toegevoegt aan de Observable Game. De GUI klasse heeft een update methode die een observable en een object binnenkrijgt en dan de GUI update door vakjes te kleuren of te melden dat een zet niet mogelijk is en dan een vakje niet te kleuren. De Game notified de GUI als het spel bijvoorbeeld gereset moet worden of als er een zet wordt gedaan. De RolitController zit hier tussen en zorgt ervoor dat wanneer er wordt gedrukt op een button op het veld, deze coördinaten worden doorgestuurd naar de Game en vice versa.

Formats voor data storage en de protcollen:

Alle data die opgeslagen moest worden, werd opgeslagen in lijsten. Zoals alle scores die worden behaald, zijn opgeslagen in een lijst. Deze lijst bestaat dan ook weer uit lijsten met individuele scores. Ook slaan we per beurt de mogelijk zetten op in een lijst. Zo kunnen deze mogelijke zetten gemakkelijk worden aangeroepen en gebruikt. Maar bijvoorbeeld alle vakjes die al zijn gevuld worden niet bijgehouden, omdat er hiervoor binnen de GUI de buttons worden gedisabled en ze niet meer terecht komen in de lijst met mogelijke zetten.
De protocollen zijn onderverdeelt in drie klasses. Er is een CommonProtocol die onder andere beschrijft hoe een versie wordt beschreven en wat er wordt ondersteund door de client of de server. Ook staan hier de printstream en de bufferedreader in. Verder is er nog een ClientProtocol en een ServerProtocol. Deze zijn specifiek gemaakt voor de client en voor de server. In het ClientProtocol zijn alle commando's en abstracte methodes te vinden die worden aangeroepen als een bepaalde commando wordt aangeroepen door de client. Voor het ServerProtocol geldt hetzelfde als voor het protocol van de client. Alleen zijn hier ook alle mogelijke errors in gedefinieerd.

Discussion per Class:

Board

De rol van de klasse Board in het systeem is dat Board ervoor zorgt dat alle regels van Rolit worden waarborgt, zoals waar de speler een kleur mag plaatsten en of de speler moet slaan, of er een winnaar is en wie dat dan is. Ook wordt er gecheckt of het bord vol is en dus ook of het spel voorbij is. Board zorgt er ook voor dat wanneer er wordt geslagen, de juiste kleuren veranderen op het bord. Board houdt ook bij wat de actuele status is van elk veld op het board. Dit wordt dan doorgegeven aan de GUI, zodat deze gelijk lopen aan elkaar.

Board is dus verantwoordelijk voor alle spelregels van het spel. En voor het controleren of het bord vol is en of er dus een winnaar is.

Voor de Board klasse is de Mark1 klasse gebruikt, zodat de Board klasse kan werken met de vier kleuren van rolit.

Waar op gelet moet worden bij het invullen van de coördinaten in bepaalde methodes van de Board klasse is dat deze coördinaten groter moeten zijn dan nul en kleiner of gelijk aan de DIM, dus kleiner of gelijk aan 8.

ClientHandler

De ClientHandler is een soort onderdeel van de server, de ClientHandler neemt als het ware de rol van de server voor deze client over zodra die client zich heeft gemeldt. De ClientHandler gaat zorgen voor de communicatie tussen de 'de server' ofwel zichzelf en de client. Dit is zodat de server zich bezig kan houden met het bijhouden van de clients en het starten van een nieuw spel mocht dat aan de orde zijn.
De ClientHandler is dus verantwoordelijk voor de communicatie. Elke client heeft een eigen ClientHandler.

ComputerPlayer

Het is een speciale vorm van Player. De Computerplayer zorgt ervoor dat er gespeeld kan worden als AI. Hier kan gekozen worden welke strategie de AI handhaaft. Als er dan geen moeilijkheidsgraad wordt gekozen, wordt automatisch gekozen voor de NaiveStrategie.

De Computerplayer bepaald een move door middel van de determineMove methode. Binnen deze methode wordt de huidige strategie gebruikt die de werkelijke zet uit rekent.
Voor de Computerplayer klasse wordt dus de huidige strategie gebruikt. Dit is of de NaiveStrategy, of de SmartStrategy of de PerfectStrategy. Verder extends de Computerplayer klasse de Player klasse. En ComputerPlayer gebruikt de Mark1 klasse om de huidige mark te bepalen, of te wel om de huidige speler te bepalen.

Er zijn binnen deze klasse geen speciale gevallen te noemen.

Wel moet er een kleur worden meegegeven, als Mark1 type. En de strategie moet mee worden gegeven als Strategy type.

Game

De Game klasse houdt bij wie er aan de beurt is, hoeveel spelers er mee spelen en hoe het bord erbij ligt. Game is de Obervable klasse. Game extends Observable Deze kan dus een spel starten, door middel van het aanmaken van een bord en de spelers. Ook kan Game het bord resetten en een zet doen als de methode 'takeTurn' wordt aangeroepen. Game is dus het *Model* in de *Model-View-Controller*.

GUI

De GUI zorgt voor de grafische interface van het spel. Dus hij zorgt ervoor dat alle 64 vakjes op het scherm komen te staan en dat de beginpositie bij de start van het spel gelijk goed staat. Daarnaast ontvangt de GUI van Game welke vakjes van kleur zijn veranderd binnen de Board klasse.

De GUI is de *View* in de *Model-View-Controller*. De GUI wordt aangeroepen door de klasse start zodra er een valide poort is ingevoerd en een naam als er een humanPlayer gaat spelen.
De GUI is verantwoordelijk voor het aanmaken van de knoppen en het enabelen en disabelen van de knoppen zodra er op de knop geklikt is, zodat er geen twee keer op de zelfde knop gedrukt kan worden. Dit doet de klasse door te kijken welke kleur de knop heeft en deze alleen enabled te maken als hij zwart of grijs is, grijs voor als er om een hint wordt gevraagd, dan kleuren de knoppen grijs. Verder is de GUI de Oberver. Deze implementeert Observer en ActionListener en extends JFrame.

De ActionListener wordt geimplementeerd om te zorgen dat wanneer de knop replay (New Game?) wordt ingedrukt er een nieuw spel wordt gestart. Dat doet de GUI zodat het rolitFrame meteen gesloten kan worden en er opnieuw Start() aan geroepen wordt. Op deze manier kun je meteen kiezen om tegen een ander aantal mensen of via een andere poort te spelen, of de AI aan te zetten.

HumanPlayer

De HumanPlayer zorgt ervoor dat de speler een zet kan doen en kijkt door middel van methodes uit Board of deze zet geldig is. Wanneer dit niet zo is, wordt dit gemeld aan de speler. Als de gekozen zet wel mag, wordt deze doorgegeven aan het bord, zodat deze zet gedaan kan worden. Dit gaat dan via de Player interface.

HumanPlayer extends de Player klasse. Dit omdat de HumanPlayer een andere Player is dan een ComputerPlayer. Verder zijn er geen klassen vereist voor de HumanPlayer.

Leaderboard

Deze klasse zorgt ervoor dat er een leaderboard kan worden bijgehouden en er scores uit het leaderboard kunnen worden opgevraagd. Zoals de top zoveel van de leaderboard, de topscores van een bepaalde dag of alle scores van een bepaalde persoon. Ook is er een methode om het leaderboard te sorteren van hoogste naar laagste score, omdat je elke nieuwe score gewoon achter bij de andere scores voegt, en niet op de juiste plaats in de lijst. Daarom is er een methode om het leaderboard te sorteren.

De Leaderboard klasse moet ervoor zorgen dat het leaderboard bestaat uit een lijst van lijsten, waarbij de laatste lijst bestaat uit een naam als String, een score als een integer, een datum als String met als volgorde dd/mm/jjjj en een tijd als String met als volgorde hh/mm/ss. Ook moet de klasse ervoor kunnen zorgen dat de leaderboard kan worden gesorteerd.
Voor de Leaderboard klasse zijn er geen andere klassen gebruikt die we zelf hebben geschreven. Wel hebben we uit de JRE System Library de LinkedList gebruikt uit java.util. De LinkedList is gebruikt om de lijst van lijsten te maken voor het scoreboard, waar dan de eerstgenoemde lijst een LinkedList van LinkedLists is en dus het werkelijke leaderboard en de laatstgenoemde lijst is een LinkedList van een individuele score.

Binnen de klasse zijn er geen speciale gevallen waar rekening mee moet worden gehouden. Alle mogelijke fouten die er kunnen ontstaan, zijn netjes afgevangen.
Voor elke query waar een waarde moet worden meegeven, moet er wel het juiste type waarde worden ingevuld. Ook moet er voor de methode voor het opvragen van de top zoveel van de leaderboard er een positieve waarde worden ingevuld.

Mark1

De rol van deze klasse is de kleuren als een enum definiëren, zodat deze als ROOD, GROEN, GEEL en BLAUW gebruikt kunnen worden als type Mark1. Ook zorgt Mark1 ervoor dat in het spel na de beurt van een speler, de volgende speler wordt gekozen aan de hand van het aantal spelers. Dit moet aan de hand van het aantal spelers, omdat de kleurvolgorde verschilt per aantal spelers.

Voor deze klasse zijn geen andere klassen gebruikt.

Verder zijn ook geen speciale gevallen te noemen voor deze klasse.

De enige vereiste in de klasse is dat het aantal spelers twee, drie of vier moet zijn, niet meer en niet minder. Dit omdat het spel niet meer of minder spelers ondersteunt.

NaiveStrategy

NaiveStrategy vraagt alle mogelijke zetten voor de kleur die aan de beurt is op bij Board, Board geeft een lijst met zetten die gedaan mogen worden. NaiveStrategy kiest een van deze zetten met behulp van Math.random zodat er een random zet gedaan kan worden.

De NaiveStrategy implementeert de Strategy interface.

Player

Player is een abstracte klasse die een nieuwe speler creëert, dit kan een Computerplayer of een HumanPlayer zijn, afhankelijk van wat er gekozen is door de speler.

De Player maakt de werkelijke move op het bord.

De klasse gebruikt de Mark1 klasse om een mark te maken, zodat te actuele mark gereturned kan worden.

Ook vraagt de Player klasse een bord op om een move op te maken.

RolitController

Deze klasse zorgt voor de werking van de knoppen op het speelbord. De klasse ontvangt van de GUI welke knop er wordt ingedrukt en geeft deze informatie weer door aan de Game waarna de de Game weer vraagt aan het Board of de zet mag of niet.

Deze klasse heeft dus de Game klasse en de GUI nodig om te functioneren. De GUI heeft hij dus nodig om te weten welke knop er wordt ingedrukt door de speler en de Game heeft hij nodig om te vertellen welke knop en moet worden veranderd, of niet. Ook heeft de RolitController klasse het board nodig om te vragen welke coördinaten de knop heeft die wordt ingedrukt, zodat dit aan Game kan worden meegegeven.

Server

Server is constant aan het wachten tot er een client is de zich aan meldt. Zodra een Client zich aanmeldt maakt server een nieuwe ClientHandler aan en laat de communicatie verder aan de ClientHandler over. Alleen als de ClientHandler iets naar alle andere clients wil sturen doet hij dit via de server omdat de server een lijst bijhoud met alle clientHandlers en dus makkelijke een bericht naar clientHandlers kan sturen.

Start

Start is echt de start van het spel. Als je Start runt krijg je een scherm te zien waarop je, door middel van een dropout scherm kan kiezen met hoeveel spelers je wilt spelen, je kunt er het

poortnummer in voeren en je kunt kiezen of je zelf wilt spelen of een AI voor jou wil laten spelen. Als je zelf gaat spelen moet je een naam invoeren en wanneer je de AI laat spelen moet je een strategie kiezen in het dropout venster. Wanneer je dan op de knop "Start Game" klikt start het spel, tenzij de poort niet goed is, dan krijg je een venster met een error, als je op dat venster op "OK" gedrukt hebt kun je een ander poortnummer invoeren en het nog eens proberen.
Start is dus verantwoordelijk voor het aanroepen van de GUI en het sluiten van zijn eigen venster zodra de GUI loopt. Deze klasse staat verder volledig op zichzelf, hij implementeert alleen ActionListener om te zorgen dat de knopjes doen waar ze voor gemaakt zijn.

Strategy:

Dit is een interface voor de strategieën voor de computer gestuurde spelers. Hier worden verder geen andere klasses voor gebruikt.

Deze klasse is verantwoordelijk voor de basisopbouw van de strategieën voor de computer gestuurde spelers. Naast NaiveStrategy heb ik nog geen strategieën kunnen schrijven omdat ik hier geen tijd voor gehad heb.

Test Report: Unit Testing

Board

Board wordt getest door de klasse BoardTest. Hiervoor wordt wordt alleen Board en BoardTest gebruikt. Dit is dus net als bij het Leaderboard een unit test. Voor BoardTest heb ik niet elke methode getest die in Board staat, dit is voornamelijk omdat die methoden later bij zijn gemaakt en BoardTest niet is aangepast, de coverage volgens EMMA is dan ook 79,9%. Nogsteeds een mooie score.

Waar BoardTest zich vooral mee bezig heeft gehouden is kijken of alle velden de juiste kleur hebben en krijgen en of het opnieuw kleuren van de velden goed gaat als er een zet gedaan is. Vooral het herkleuren van de velden is zeer uitgebreid getest. Later, toen ik klaar was met de GUI en het spel als Stand Alone wilde gaan spelen kwam ik erachter dat er nog stukjes niet helemaal werkte. Waar ik op had getest was in de binnen kant van het bord, dus tegen het midden aan. Ik had dus niet getest of het ook nog werkte aan de randen, dit deed het helaas niet helemaal. Dat heeft me later dus nog tijd gekost om ook dat werkend te krijgen.

BoardTest is dus eigenlijk een beetje achterhaald, maar het systeem test board zeer goed, vooral met de GUI is Board erg goed te testen. Ook de hint knop is een extra test voor de activeField methode van Board.

ClientHandler

ClientHandler wordt door het systeem getest, verder is er geen test voor ClientHandler geschreven. Deze klasse is zelf ook nog niet af wegens tijd gebrek dus werkt ook nog niet naar behoren. ClientHandler wordt vooral getest door Start, en de regels die door Start worden geprint.

ComputerPlayer

ComputerPlayer wordt, wordt nog niet gebruikt, omdat de server nog niet af is. Hierdoor wordt ComputerPlayer niet door het systeem getest. Ik heb ook (nog) geen unit test geschreven voor deze methode.

Game

Game wordt alleen door het systeem getest in game zijn een heel aantal methoden nog niet gebruikt. Deze gaan gebruikt worden door Player. Deze methodes zijn dus ook nog niet getest. Game is voornamelijk verantwoordelijk voor het opvragen van een lijst met mogelijke zetten voor hint, voor het maken van een zet, dus de methode take turn en zorgen dat er een volgende kleur gespeelt gaat worden door currentMark aan te passen als er een zet gedaan is. Deze methodes worden getest in de GUI en werken.

GUI

De GUI test zichzelf voornamelijk en test zelf veel dingen in het de rest van het systeem. De GUI is te testen door te kijken of de verschillende knoppen het op het juiste moment wel of niet doen, door te kijken of je niet bijvoorbeel een zet kan doen op een al gekleurd veld, of er niet een nieuw spel gestart kan worden als het huidige spel nog niet is afgelopen enzovoort.

HumanPlayer

Bij HumanPlayer is het zelfde verhaal als bij ComputerPlayer. Voor HumanPlayer is nog geen unit test geschreven en HumanPlayer wordt ook niet door het systeem getest omdat HumanPlayer nog niet gebruikt wordt omdat de server nog niet klaar is en ik geen tijd heb gehad HumanPlayer al in het geheel op te nemen.

Leaderboard

Tijdens het testen van het leaderboard is er geen andere klasse in de test gebruikt dan de leaderboard klasse.
Om het leaderboard te testen heb ik elke methode apart getest. Per methode heb ik een nieuw leaderboard aangemaakt waar ik de parameters die ik wilden testen heb gevarieerd. Zo hebben ik als ik alle scores voor een persoon wil, heb ik alleen de naam en wat scores veranderd, maar de rest gewoon hetzelfde gelaten, zodat alleen getest wordt wat getest moet worden.

Per methodetest is het leaderboard gereset en opnieuw gevuld met scores. Zo kunnen er geen complicaties optreden en wordt elke methode echt afzonderlijk van de rest getest. Daarom wordt er ook voor elke test een nieuw leaderboard aangemaakt en worden alle scores die nog aanwezig zijn verwijdert door een leaderboard.reset(). Deze methode staat in de leaderboard klasse zelf en gooit de lijst waar één score van één persoon instaat leeg en de lijst waar alle scores instaan wordt dan leeggegooid.

Voor de leaderboardtest is er volgens EMMA een 100% coverage van de leaderboardklasse. Dit betekend dat elke mogelijkheid die binnen de klasse kan voorkomen is getest en dat er geen onverwachte dingen kunnen gebeuren als de leaderboard klasse wordt uitgevoerd. De leaderboardtest test dus alle methodes uit leaderboard op alle mogelijke manieren.

Mark1

Mark1 word getest door de Mark1Test. De coverage is 83.4%. De enige dingen die hij niet zegt te testen test hij weldegelijk dus dat is wel raar. Ik heb de test zo geschreven dat hij met 3 verschillende methoden test, elke methode kijkt of de vershillende kleuren de juiste kleur worden voor een bepaalde waarde van i.

De Mark1 next methode wordt ook getest door het systeem. Want in de start wordt een waarde mee gegeven door te kiezen met hoeveel het spel gespeeld gaat worden. Deze waarde wordt opgeslagen in Game en Game roept de next functie van Mark1 aan en vult hier de waarde op in om de volgende kleur op re vragen aan Mark1. In de GUI kun je zien dat dit werkt. Als je bij een de stand alone game met twee spelers speelt krijg je alleen de kleuren rood en groen en met drie spelers krijg je rood, geel en groen en bij vier spelers krijg je ze alle drie.

NaiveStrategy

NaïveStrategy wordt niet getest door het systeem en er is ook nog geen unit test geschreven voor NaiveStrategy omdat hier geen tijd voor was. Dat NaiveStrategy ook niet wordt getest in het systeem is omdat NaiveStrategy nog niet is opgenomen in het systeem. Wel is de kans dat NaiveStrategy werkt erg groot, aangezien NaiveStrategy in grote lijnen het zelfde werkt als hint, alleen wordt er nu een random veld uit de lijst gekozen inplaats van dat alle velden in de lijst gekleurd worden. Aangezien de hintknop erg goed werkt zie ik geen rede dat de NaiveStrategy niet zou werken.

Player

Player wordt nog niet gebruikt in mijn implementatie en dus ook niet getest door het systeem. Ook heb ik geen unit test geschreven voor Player. Player wordt dus niet getest.

Rolit

Rolit zal de players gaan creeëren. Aangezien ik er nog niet aan toe ben gekomen de players in het spel te verwerken heb ik ook rolit nog niet nodig gehad. Rolit wordt dus niet door het systeem getest en er is ook nog geen unit test voor rolit geschreven wegens tijdgebrek.

RolitController

RolitController is een klasse die zorgt voor de communicatie tussen de GUI en de klasse Game. RolitController wordt dus door het systeem getest, maar nog niet door een unit test. Het werken van de Stand Alone rolit is eigenlijk de volledige test van de RolitController.

Server

De server wordt getest door de methode Start en door de ClientHandler. Het testen van de server bestaat dus uitsluitend uti systeemtests. Een unit test is hier (nog) niet voor geschreven aangezien de server zelf nog niet af is.

Start

De Start klasse is eigenlijk een ClientGUI en test zichzelf dus voornamelijk. Of deze klasse werkt of niet is makkelijk te zien door te kijken of de verschillende knoppen en fucnties wel of niet werken, zo kun je proberen geen naam in te voeren, dan komt er een popup venster wat verteld dat de username niet juist is, ook als er een ongeldig poortnummer wordt ingevoerd verschijnt er een popup, dit is dus voornamelijk zelf te testen.

Reflection on planning of the project week:

Mijn ervaring in week 4

Mijn ervaring met het tijdschrijven in week 4 is vrij goed, ik heb mijn planning gemaakt met de ervaringen die ik heb met hoe ik de afgelopen weken heb gewerkt en wat ik heb gedaan. Mijn beeld van de hoeveelheid werkuren die ik maak in de week klopt met de hoeveelheid tijd die ik daarvoor heb ingepland. Als ik een planning zou maken van hoe ik vind dat een week eruit zou moeten zien, zou dat waarschijnlijk wel anders zijn geweest in week 4, omdat ik daar nogniet bezig was met project. In de weken die ik heb besteed aan project heb ik meer tijd besteed aan mijn studie dan gepland.

Ik merk eigenlijk vooral dat de volgorde waarin ik de dingen heb gedaan verschillen, en niet de hoeveelheid tijd die ik eraan besteed heb. Soms schuif ik dingen op naar een andere dag, of naar een ander moment op de dag, maar ik doe ze wel.

Projectweek planning

De planning in de projectweken was vrij druk. We hadden geplant om van 9 uur 's morgens tot 7 uur 's avonds te werken, 2 uur per dag zouden we besteden aan het leren van de herkansingen, de rest van de tijd was voor project. De meeste dagen ven ik om een uur of half 8, 8 uur al aan het werk gegaan en we zijn dan door gegaan tot een uur of 11 's avonds, dan waren we het echt zat en gingen slapen. Ook het voornemen om 2 uur per dag aan de herkansingen te leren is er eigenlijk elke dag bij in geschoten.

De rede dat we van onze planning afweken

De rede dat we van onze planning zijn afgewijkt is omdat het al snel duidelijk werdt dat we nog heel veel moesten doen. In de vakantie was ik bezig geweest met het afmaken van wat afteken opdrachten. In de week na de vakantie heb ik dus de Board klasse geschreven, hierdoor heb ik weinig tijd besteed aan bijvoorbeeld mijn wiskunde. Aan het einde van de eerste week na de vakantie was de klasse Board en zijn test af. In het weekeinde dat daarop volgde voorzag ik wel al dat het nog heel veel harder werken zou worden dan dat ik toen al had gedaan, die week had ik al nogal veel tijd besteed aan het programmeren, omdat het zo lekker ging met het board. Dat is vanaf toen niet meer veranderd.

Ik heb dus de afgelopen 4 weken eigenlijk nonstop aan programmeren gewerkt, nog even in een weekeinde, mijn toets geleerd, en weer verder aan project. Ik heb elke dag veel langer door gegaan dan gepland, en dan de volgende morgen weer eerder begonnen, om het af te krijgen maar dat wilde niet lukken. Ik heb zeker veel vooruitgang geboekt, op een gegeven moment was de GUI helemaal af en moest er alleen nog een server komen. Vooral de server werkend krijgen kostte veel meer tijd dan gedacht, daardoor heeft Gerwin, die dit zou doen, verder weinig aan het project gedaan. De server die nu bij het project zit is niet van Gerwin, die heb ik nog gebouwd in het laatste weekeinde.

Eigenlijk kostte alles meer tijd dan gedacht door het gebrek aan ervaring met programmeren is er veel tijd gaan zitten in het opzoeken van hoe de dingen nou moeten werken en het zoeken naar foutjes die hier en daar on het het project zaten. Ik vind dat erg jammer dat ondanks mijn harde werk het project niet is afgekomen want ik had het project graag af gezien en ik heb ook heel erg hard gewerkt om het af te krijgen.

Vooral met het maken van de server is voor Gerwin veel tijd verloren zonder veel resultaat. Ook het begrijpen van de *Model-View-Controller* heeft meer tijd gekost dan nodig was. Dit was vooral omdat zowel hij met de server als ik met de MVC geen hulp zochten, dat heb ik na anderhalve dag klooien wel gedaan, omdat ik verder wilde, Gerwin niet, waardoor de server waar hij mee bezig was nogsteeds eigenlijk niets doet.

Andere tijdrovende dingen

Wat impact heeft gehad op de tijd die we voor het project hebben gehad zijn de herkansingen die we moesten doen. We hebben allebij nog herkansingen moeten doen om te zorgen dat we de toetsen goed gaan afronden. We hadden niet verwacht zo veel toetsen over te hoeven doen. Door de druk van het project heb ik helaas ook niet voldoende tijd aan mijn (her)tantamens kunnen besteden, waardoor die ook niet zijn gegaan zoals mogelij was geweest als ik hier meer tijd aan had besteed. Dus het is eigenlijk niet zo dat het leren van de herkansingen veel tijd heeft gekost, maar wel een beetje, en in je hoofd ben je er toch mee bezig. Verder is het maken van de herkansingen zelf wel tijdrovend, daar ben je wel een halve dag aan kwijt.

Dingen die we hebben geleerd over planning

Onze planning opzich was niet heel slecht. We zouden voor een volgende keer meer reserve tijd kunnen inroosteren als er weer onverwachte herkansingen komen, maar ook als we geen herkansingen gehad hadden hadden we toch nog te weinig tijd gehad. We zijn ook niet te laat begonnen met werken of hebben dingen lang laten liggen. We hebben We hebben namelijk niet in het begin rustiger aan gedaan omdat we het project onderschat hadden en dus rustiger aan te kunnen doen. Ik kan wel zeggen dat ik deze hele module echt te hard gewerkt heb.

Ik denk dat dit het eerste project is in mijn leven dat ik niet heb uitgesteld maar waar ik meteen aan begonnen ben, dat voelt wel fijn, het idee dat het in ieder geval niet aan jou ligt dat het niet af is, dat je er echt alles aan hebt gedaan wat er maar in je macht lag om het af te krijgen. Dan heb je in ieder geval niet het schuld gevoel wat je nog al eens hebt naar jezelf toe als je aan het einde van een project veel te hard moet werken om het af te krijgen terwijl je er in het begin met de pet naar gegeooid hebt.

Wat ik andere zou adviseren

Vraag optijd naar hulp! Je hebt echt geen tijd om zelf bezig te zijn met uitzoeken hoe het nou precies moet dus ga naar mensen toe die je kunnen helpen. Er als je het hebt gevraagd achterkomen dat het eigenlijk heem makkelijk was is minder vervelend dan heel lang zelf proberen een oplossing te vinden en dan heel veel tijd kwijt zijn met zoeken.

Verder zou ik adviseren om het werk goed te verdelen en van tevoren goede afspraken te maken over wie wat gaat doen, zodat niet de een heel hard gaat werken, en dan daarmee ook de ander belemmerd in wat hij nog kan doen, dat is een van de diengen die bij ons fout ging, doordat ik aan het werk was gegaan snapte ik wat er allemaal gedaan moest worden, dan de ander uitleggen wat die moet gaan doen kost dan naar je idee zoveel tijd dat je het dan liever gewoon zelf doet en je zeker weet dat het ook gebeurt zoals je het in je hoofd hebt.

Verder zijn de standaard dingen zoals optijd beginnen en geen dingen voor je uit schuitven een goede tip. En natuurlijk goed in de gate houden wat er allemaal gedaan moet worden, zodat je niet vergeet dat er bijvoorbeeld nog een verslag gemaakt moet worden.

Nawoord

Ik vond het echt geweldig om bezig te zijn met het programmeren! Helaas heb ik er alleen niet heel vaak van kunnen genieten omdat het zo veel werk was en er nog zo veel moest gebeuren en ik wel door had, aan het begin al dat dat heel hard door werken zou gaan worden.

Ik had mijn spel zo ontzettend graag af gezien en ik had er zo graag iets heel moois van willen maken! Ik zou zo trots geweest zijn als het was gelukt om het af te krijgen. Zo trots dat het mij, die nog niet kon programmeren voor ik begon aan deze opleiding, samen met iemand die dat evemin kon, tegen iedereens verwachtingen in het was gelukt om heel rolit af te krijgen. Dan had ik kunnen zeggen: "zie je wel dat ik het wel kan", tegen iedereen die had gezegt dat dat niet kon en dat ik maar met iemand moest gaan samenwerken die al beter kon programeren omdat het anders niet mogelijk was om het af te krijgen. Ik heb ook geprobeerd iemand te vinden die wel al kon programmeren, maar iedereen had eigenlijk al iemand. Niet dat ik het niet leuk vond om met Gerwin samen te gaan weken, alleen was het gewoon veel te veel voor een groepje zonder programmeer ervaring.

Achteraf heb ik er geen spijt van dat ik niet met iemand samen heb gewerkt die al beter kon programmeren, ik heb echt enorm veel geleerd, zoveel had ik niet kunnen leren als ik samen had gewerkt met iemand die al beter kon programmeren, omdat je dan de moeilijke dingen toch overlaat aan de ander en zelf de stukjes gaat doen die wat makkelijker zijn.

Ondanks dat het me niet is gelukt om het spel af te krijgen ondanks dat ik zoveel tijd erin heb gestoken dat het verslag goed uitgewerkt is en ik me goed voor te kunnen bereiden op mijn toetsen weet ik wel dat ik heel veel geleerd heb, veel meer dan ik had kunnen doen als ik met iemand anders had samen gewerkt en dat ik, of ik de module nu ga halen of niet, er iets is wat mij deze ervaring en kennis nog kan afnemen. Ik ben enorm veel ervaring en ook veel kennis rijker en heb daarbij ook nog gemerkt dat ik deze opleiding echt leuk vind, ondanks de stress en de vele uren frustratie die het me heeft gekost, voornamelijk toen ik merkte dat ik het nooit af zou gaan krijgen.

Ik hoop natuurlijk enorm dat ik deze module wel gehaald heb, ondanks alles, maar ook als dat niet zo is blijf ik met trots terug kijken naar wat ik berijkt heb en naar wat ik allemaal geleer heb en wat ik echt op eigen kracht voor elkaar heb gekregen. Gewoon omdat ik met trots kan zeggen dat ik het echt zelf gedaan heb.