CDIO - 2

Et simpelt brætspil for to spillere.

Af gruppe 33

Simon Engquist
s143233

Arvid Langsø
s144265

Mikkel Lund
s165238

Jeppe Nielsen
s093905

Mads Stege
s165243

Danmarks Tekniske Universitet
DTU Compute

04. november 2016 - Kl. 23:59

Side antal (med bilag): 24

Kurser: 02312, 02313, 02315

CDIO - 2							
Time-regnskab	Ver. 2014-11-11	Tidsplan for CDIO del 2 - 2016					
Dato	Deltager	Design	Impl.	Test	Dok.	Andet	Ialt
2016/10/10	Mads	2.5					2.5
2016/10/10	Jeppe	2					2
2016/10/10	Simon	2					2
2016/10/10	Mikkel	2					2
2016/10/10	Arvid	2					2
2016/10/11	Mads	2					2
2016/10/11	Jeppe	2					2
2016/10/11	Simon	2					2
2016/10/11	Mikkel	2					2
2016/10/11	Arvid	2	1				3
2016/10/13	Mads		6.5				6.5
2016/10/13	Jeppe		6.5				6.5
2016/10/13	Simon		6.5				6.5
2016/10/13	Mikkel		6.5				6.5
2016/10/13	Arvid		7.5				7.5
2016/24/10	Mads				2		2
2016/24/10	Simon				2		2
2016/24/10	Mikkel				2		2
2016/27/10	Mads			4	2		6
2016/27/10	Jeppe			4	2		6
2016/27/10	Mikkel			2.5	3.5		6
2016/27/10	Arvid			2.5	3.5		6
2016/28/10	Mikkel				1.5		1.5
2016/28/10	Mads				1.5		1.5
2016/28/10	Arvid				1.5		1.5
2016/28/10	Jeppe				1		1
2016/31/10	Mikkel				2		2
2016/31/10	Mads				2		2
2016/31/10	Arvid				2		2
2016/31/10	Jeppe				2		2
2016/31/10	Simon				2		2
2016/04/11	Mikkel				7		7
2016/04/11	Mads				7		7
2016/04/11	Arvid				7		7
2016/04/11	Jeppe				7		7
2016/04/11	Simon				7		7
	Sum	20.5	34.5	13	67.5	0	135.5

Indholdfortegnelse

1 Abstract **4**

2 Indledning **4**

3 Problemformulering **4**

4 Krav Specifikation **5**

5 Design **6**

6 Implementering **13**
6.1 Minimumskrav og installationsguide 15

7 Test **16**
7.1 Acceptance test 17

8 Forbedringsforslag **18**

9 Konklusion **19**

10 Bilag **19**
10.1 Formelle Test 19

1 Abstract

This project is a continuation of our previous dice-game project. This new project aims to create a competitive board game.

A key component in creating an immersive gaming experience is done by making an interesting story, or myths. A dice game offers little opportunity to expand, in terms of game lore. Tasked with creating a board-game, with parts from the previous dice game, a much greater way of drawing in the users comes forth, and conceiving an interesting game universe is made easier.

This project has birthed a board game for two players, called ”A Tale of Tales”. The objective is rather plain. Gather 3000 points to win, achieved by landing on fields on a digital board. The games mechanics are simple, but provide an adequate environment for writing compelling fragments of story texts, making for a better experience for the user.

2 Indledning

Brætspil er et af de ældste underholdningsmetoder, og kan findes i mange varianter. Et brætspil kan opstille mange forskellige miljøer og baggrundshistorier.

Det er blevet besluttet at det færdige brætspil skal skrives i Java kode, og skal foregå i et fantasi-univers.

Der anvendes segmenter fra det tidligere projekt, navnligt en Dice-class. Man benytter også en lignende kode-struktur, med et TUI. I dette projekt følger der dog også en GUI med, som spillet skal kunne sende data til. Denne GUI vil så give en grafisk repræsentation af de data. Man udvikler så at sige et mere avanceret spil, ved at bruge grundsten fra tidligere, simplere projekter.

3 Problemformulering

Målet i dette projekt er at lave koden til et brætspil. Kunden forventer et brætspil for to personer. Kunden har udarbejdet en liste over felterne som spillerne kan lande på, og disse skal implementeres i spillet. Én af de to spillere starter spillet ved at slå med terningerne. Alt efter terningernes værdier, lander spilleren på det relevante felt. Feltet har en effekt på spillerens pengebeholdning, og det videre spil. Spillets GUI skal præsentere et stykke tekst når spillerne lander på et felt, som kort beskriver handlingen, og hvordan dette præcist påvirker spilleren.

Spillerne skal starte med en pengebeholdning på 1000, og spillet skal slutte når én af spillerne når 3000 penge.

Spillet skal let kunne oversættes til andre sprog, og det skal være muligt at skifte til andre terninger.

Kunden forventer samtidig at der efterfølgende skal kunne arbejdes videre på systemet, og at systemet er grundigt testet. Der skal være dokumentation for disse tests, samt mulighed for at

kunden selv kan gennemgå disse. Kunden forventer til sidst en beskrivelse af minimumskravene til systemet.

4 Krav Specifikation

Følgende liste over alle krav ifm. projektet, udarbejdet ud fra en navneordsanalyse, på kundens egen vision af det endelige produkt.

Ikke Formelle Krav

- IF1: Spillet skal kunne spilles på maskinerne i DTU's databarer.
- IF2: Der må ikke være bemærkelsesværdige forsinkelser.
- IF3: Spillet skal nemt kunne oversættes til andre sprog.
- IF4: Det skal være let at skifte til andre terninger.

Formelle Krav

- F1: Der skal være to spillere.
- F2: Spillerne skal på skift slå med to terninger, med en samlet sum mellem 2-12.
- F3: Spillerne skal lande på felterne svarende til summen af terningerne i deres terningslag.
- F4: Feltlisten skal være som følger:

Feltliste	**Effekt**
1: Tower	250
2: Crater	-100
3: Palace gates	100
4: Cold Desert	- 20
5: Walled city	180
6: Monastery	0
7: Black cave	70
8: Huts in the mountain	60
9: The Werewall	- 80 + Extra turn
10: The pit	- 50
11: Goldmine	650

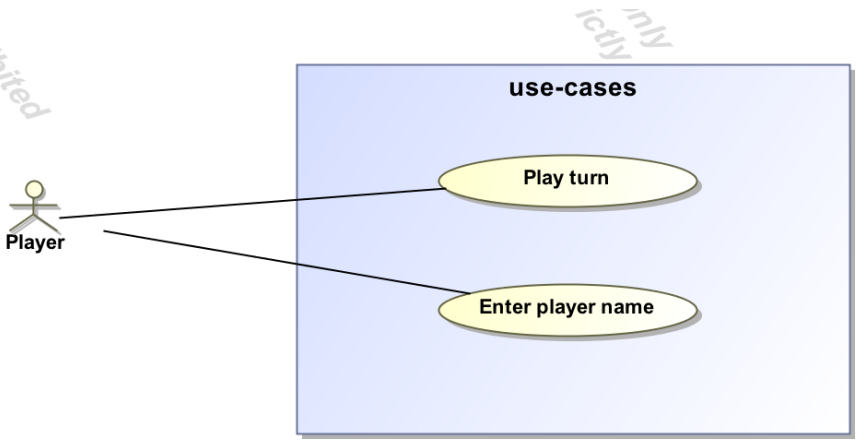
- F5: Felterne har en effekt på spillerens pengebeholdning, som anført i feltlisten.
- F6: Brættet skal udskrive en tekst, som skal beskrive feltet som spilleren lander på.
- F7: En spiller skal starte med en pengebeholdning på 1000.

- F8: Spillet skal slutte, når en spiller har en pengebeholdning på 3000.
- F9: Pengebeholdningen skal ikke kunne blive negativ.

5 Design

Alle diagrammer kan findes i mappen ”Diagrammer” på github.

Use-case diagram



Figur 3: Use-case diagram der viser de to mulige use-cases.

Use-case diagrammet viser de to use-cases, som spilleren kan gøre med brugerinterfacet. Den første use-case er at spillerne skal indtaste deres navne. Den anden use-case er at spillerne skal spille en tur. Spilleren slår med terningerne og lander på et felt. Spillerne får point svarende til hvad der står på feltet.

Use-case beskrivelser

Følgende er detaljerede beskrivelser af use-cases.

Use-case: Play Turn

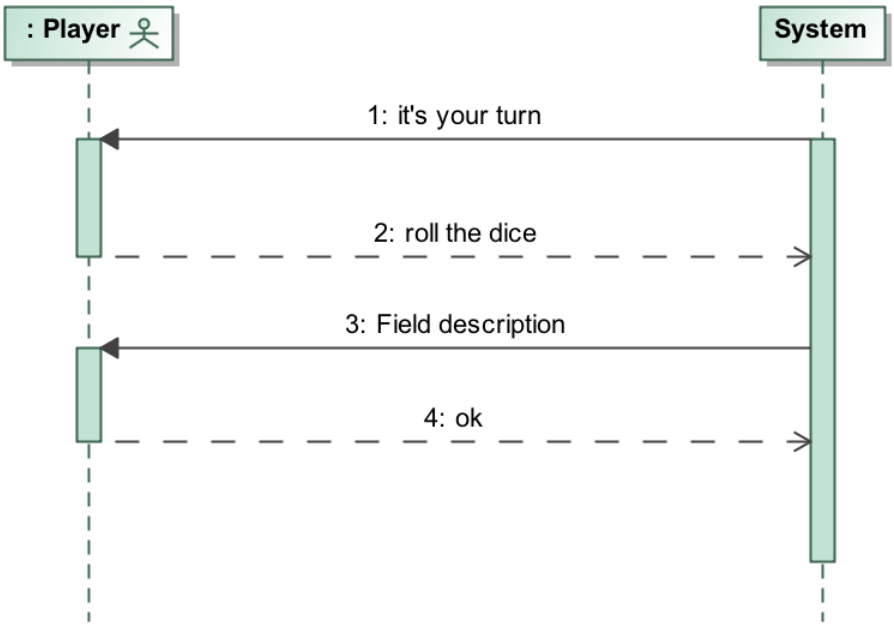
1. Systemet fortæller spilleren at det er hans tur.
2. Spilleren får tid til at læse teksten, og kan dernæst trykke ”roll”. Han har ikke andre muligheder.
3. Spilleren får besked om hvad han har slået, og hans brik bliver rykket over på feltet svarende til terningens værdi.

4. Spilleren får en tekst der beskriver hvad der sker.
   (a) Hvis han får point bliver det tilføjet til hans konto
   (b) Hvis han mister point fjernes de fra hans konto.
   (c) Hvis han er landet på werewall, får han også ekstra slag.
5. Spilleren trykker ”OK”for at færdiggøre sin tur.

Use-case: Enter Name

1. Spilleren får at vide han skal indtaste sit navn.
2. Spilleren indtaster sit navn.

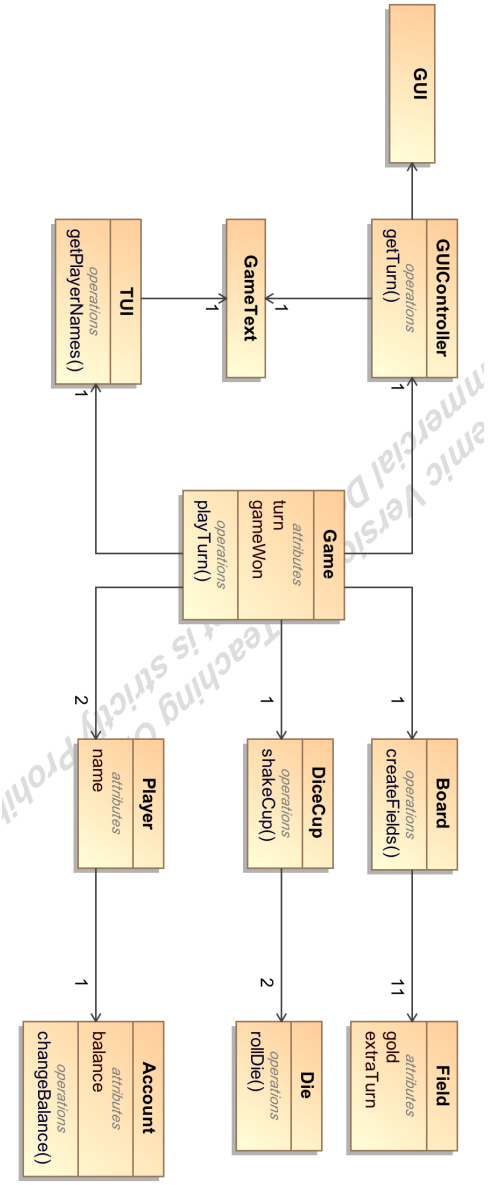
System Sekvensdiagram



Figur 4: System sekvensdiagram der viser et eksempel på hvordan spilleren kommunikerer med systemet, når han skal spille en tur.

System Sekvens Diagrammet (SSD) beskriver hvordan spilleren spiller en tur. Først fortæller systemet, at det er spiller x’s tur. Spilleren skal klikke på ”roll” for at slå med terningerne. Herefter benytter systemet spillerens feedback (”roll”) til systemet, for at hente informationerne tilhørende feltet og printe dem til spilleren. Til sidst slutter spilleren sin tur ved at klikke på ”Ok”. Det er nu den anden spillers tur.

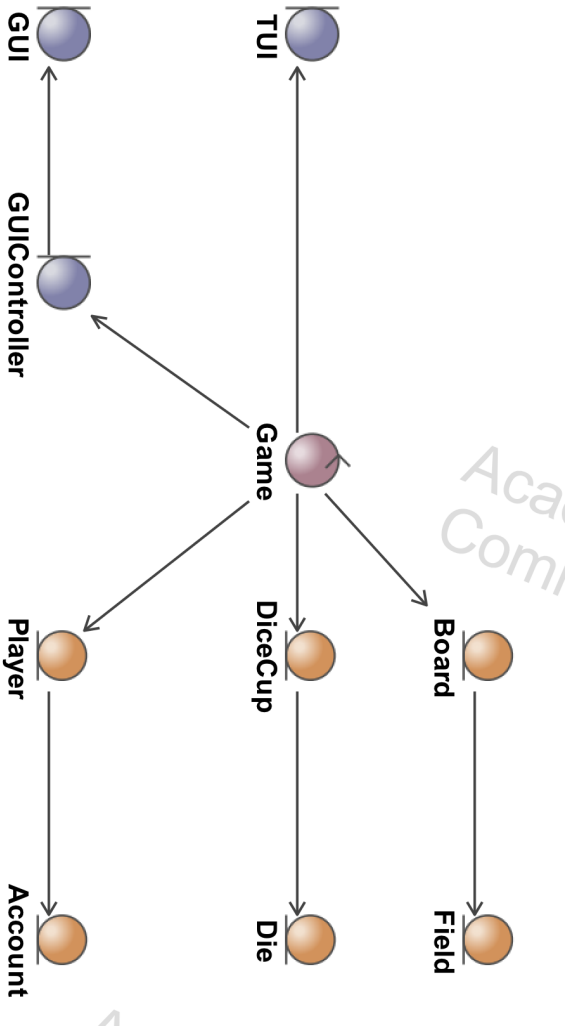
Domænemodel



Figur 5: Domænemodel

Det er besluttet at lave 10 klasser udover GUI'en, som der fik leveret. Game som har main metoden styrer spillet. Game har blandt andet en TUI og en GUIController, hvor TUI håndterer input og output fra konsollen og GUIController håndterer den grafiske interaktion med brugeren. Game har også et Board, som indeholder alle de felter (Field objekter), som spillet har. Ydermere har Game en DiceCup, som indeholder to terninger (Die objekter), som spillet skal benytte til at flytte rundt på spillerne. Hvilket til sidste bringer os til at Game også har en Player, som har en konto (Account objekt), som håndterer spillerens Score (balance).

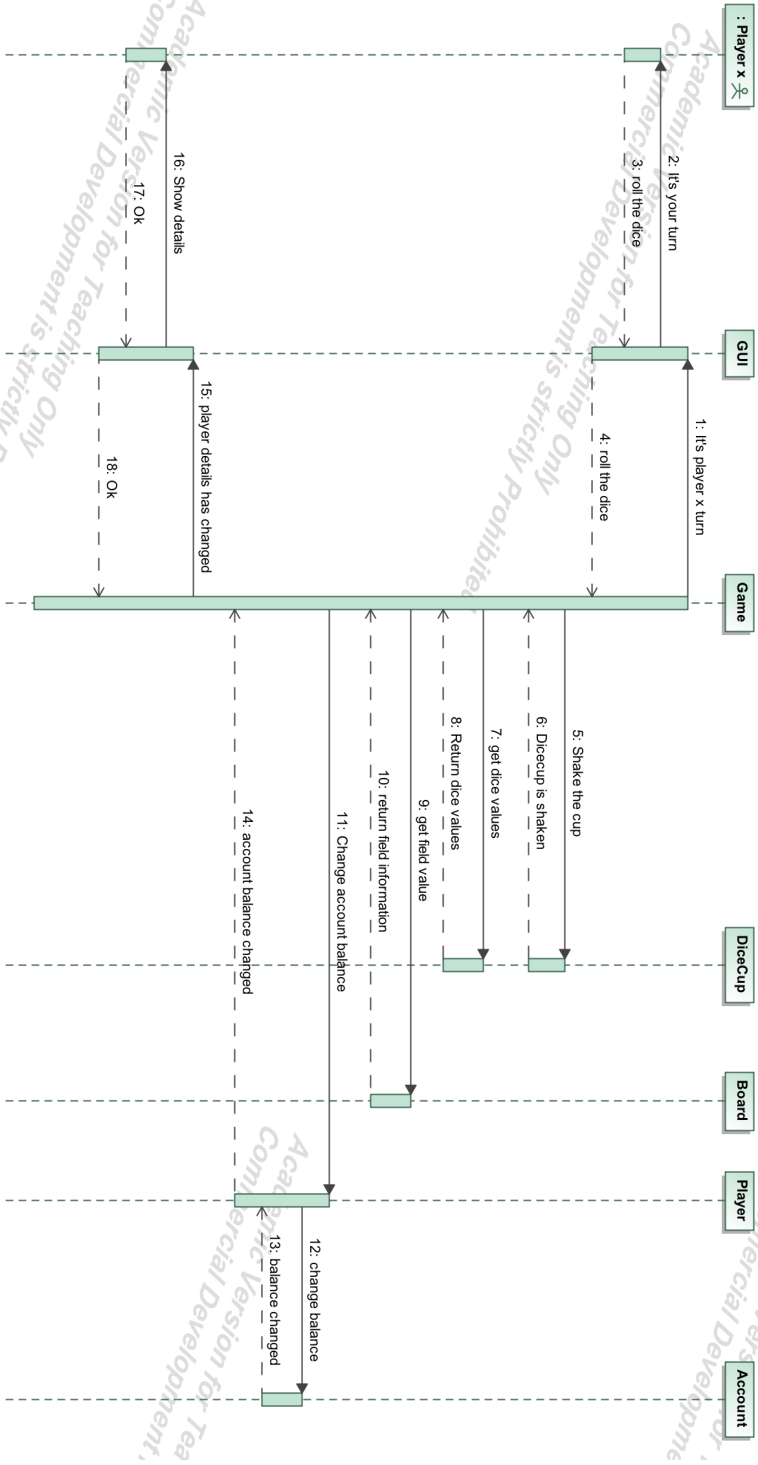
BCE-model



Figur 6: BCE model.

Denne BCE-model (Boundary Control Entity-model) viser interaktionen mellem de forskellige segmenter i dette program. Game fungerer som controller, der styrer interaktionen mellem brugerens input (håndteret af boundary klassen TUI) og det grafiske input (håndteret af boundary klassen GUIController). Derudover er der seks entity klasser, Player som har Account, DiceCup som har to Die og Board som har 11 Fields. Alle seks entity klasser leverer information til Game.

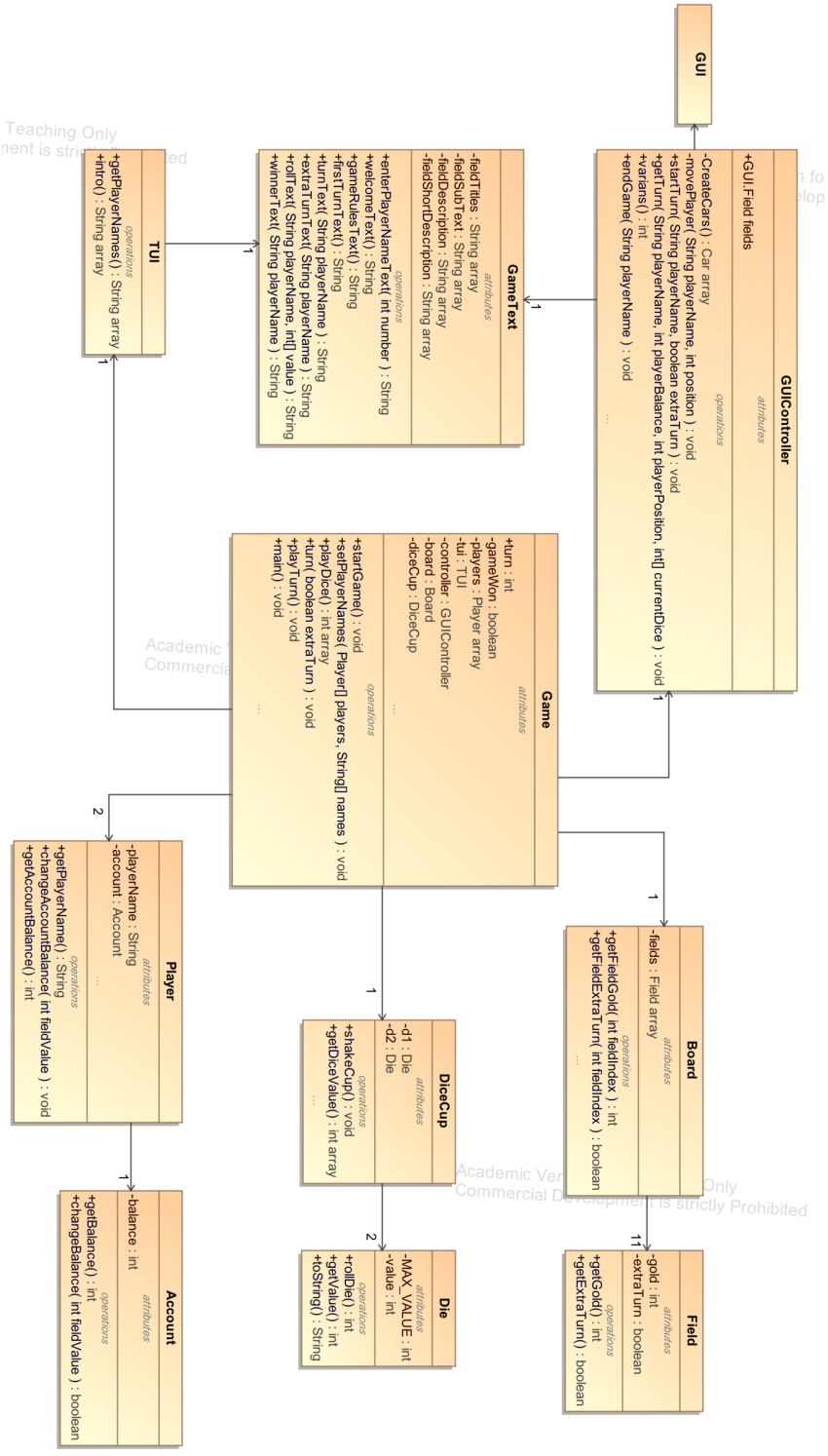
Sekvensdiagram



Figur 7: Sekvensdiagram.

- **Sekvens 1-4**: Snakker Game med GUI og aktør x for at starte en tur, og afventer ”roll the dice”fra aktør x via GUI.
- **Sekvens 5-10**: Ryster raflebæret og rykker spilleren til hans nye plads, afgjort af spillerens slag.
- **Sekvens 11-14**: Ændre spillerens balance, afhængig af hvor spilleren står.
- **Sekvens 15-18**: Viser resultatet til spilleren, via GUI’en.

Klassediagram.



Figur 8: Klassediagram

Overstående klassediagram viser de 10 klasser som er i programmet. Disse er GUI, GuiController, Gametext, TUI, Game(main), Board, Field, DiceCup, Die, Player og Account. Game

indeholder main-metoden, som igen indeholder metoderne startGame, turn og playTurn. Metoden startGame afventer to spillernavne, indtastet i TUI. Derefter kører metoden turn så længe gameWon er false, altså at "!gameWon"gælder.

Metoden playTurn som slår med terningerne(Die) i raflebæret(DiceCup), læser feltets værdi, ændrer spillerens(Player) balance(Account), kalder GUIControlleren som fjerner bilen fra det forrige felt(Board(Field)) fra hvor den stod og indsætter bilen på den nye plads.

Til sidst afventer playTurn-metoden igen, så længe ingen af spillerne har opnået 3000 point. Når "GUIControlleren"bliver kaldt, tager den fat i tekst(GameText), og derefter snakker med "GUI". Når "TUI"bliver kaldt tager den fat i tekst(GameText), afventer to spillernavne og retunerer det til "Game".

Man har prøvet at opretholde GRASP. der er derfor lav kobling mellem klasserne, hvilket fremgår ret klart her. Kun Game har en høj kobling, men man mener det er nødvendigt da den skal styre spillet. Det er blevet forsøgt at lave høj sammenhørighed, så klasserne kun har få ansvar. Fx styre Game spillet og GUI Controller skriver til spilleren. Men de har ikke andet ansvar.

6 Implementering

Følgende dokumenterer hvordan spillet er fremstillet.

Overblik

Koden er blevet implementeret, og der er brugt en blanding af konsol-inputs og en vedlagt GUI fra DTU. Når man starter spillet bliver man mødt af konsollen, der beder en om at indtaste navne på de to spillere. Derefter fortælles reglerne og GUI'en bliver åbnet.

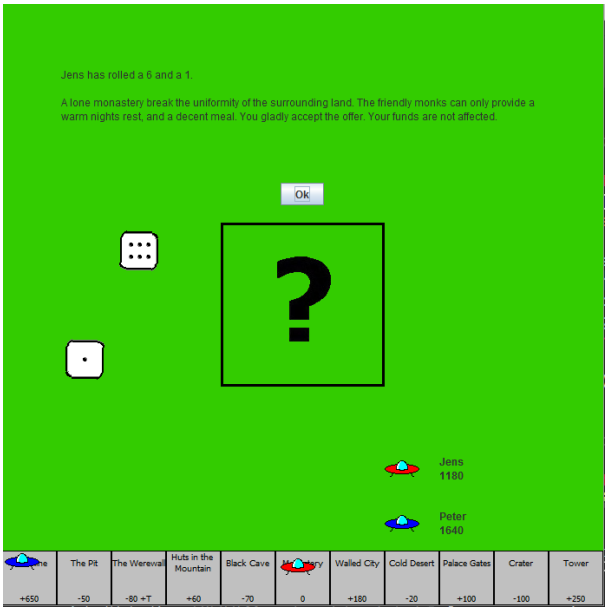
```
Welcome to 'A Tale of Tales'

Please enter the name of player 1
Peter
Please enter the name of player 2
Jens
The game rules are as follows:
- All players start with 1000 gold coins.
- The first player to achieve 3000 gold coins wins the game.
- A players funds are affected each turn. The amount depends on which field you land on in the current turn.
- A player always lands on the field number corresponding to the sum of the dice.

A player has been randomly selected to start the game.
```

Figur 9: Konsol outputtet med et eksempel outputtet i starten af konsollen.

Når navnene er indtastet kommer GUI'en frem og der kan man se spillernes point og deres placering. Der gives også besked om hvis spillers tur det er og hvilke felter der landes på.



Figur 10: Et billede af GUI'en midt i et spil.

Når spillet er vundet, så får spillerne en besked om dette.

Klasserne og deres funktioner

- **Game** - Game er main klassen, den styrer spillets logik og sørger for at spillet køres som det forventes.
- **Player** - Player indeholder information om spillerens navn, og så har den et objekt Account, der beskriver spillerens penge beholdning.
- **Account** - Account har en balance der giver udtryk for spillerens score.
- **Board** - Board indeholder 11 Field objekter, der beskriver de felter som er i spillet.
- **Field** - Field indeholder information om den mængde guld en spiller får af at lande på et felt. Den har også en boolean der fortæller om feltet giver et ekstra slag eller ej.
- **Dicecup** - Dicecup er et raflebæger der indeholder to terninger (Die).
- **Die** - Die beskriver en terning med et angivet antal sider og en nuværende værdi.
- **TUI** - TUI kommunikerer med spillerne gennem konsollen og kan bede dem om at indtaste deres navne.
- **GUIController** - GUIControlleren styrer kommuninaktion mellem Game og Klassen GUI som er givet af DTU. GUI Controlleren fortæller GUI hvad den skal vise på skærmen ud fra hvad game giver af input. GUIControlleren kan bede om at få rykket spillerne og ændre deres point, den kan desuden få GUI til at vise ting på skærmen.
- **GameText** - GameText indeholder samtlige tekster der ses af spilleren i spillet. Det er derfor nemt at oversætte spillet.

Følgende kode viser main metoden i game.

```
89          public static void main(String[] args) {
90                  //Initialize the game
91                  startGame();
92                  //Makes sure someone gets information to start.
93                  turn(false);
94                  //Game loop
95                  while (!gameWon) {
96                          playTurn();
97                  }
98                  controller.endGame(players[turn].getPlayerName());
99          }
```

Spillet starter med en metode startgame der initialiserer alle de variable. Dernæst begynder loopet der bliver ved med at spille ture indtil spillet er vundet. Playturn metoden kaldes i hver iteration. Det er klart at playTurn skifter spiller efter hver iteration, medmindre at der gives en ekstra tur. Playturn ser således ud:

```
74          public static void playTurn() {
75                  int[] currentDice = playDice();
76                  int sum = currentDice[0] + currentDice[1];
77
78                  players[turn].changeAccountBalance(board.getFieldGold(sum -
                        2));
79                  controller.getTurn(players[turn].getPlayerName(), players[
                        turn].getAccountBalance(), sum, currentDice);
80                  if (players[turn].getAccountBalance() == 3000) {
81                          gameWon = true;
82                          return;
83                  }
84                  turn(board.getFieldExtraTurn(sum - 2));
85          }
```

Først slås der med terningerne. Dernæst ændres spillerens balance, så han ender med de point spilleren skal have. Dernæst bliver GUI'en bedt om at vise alt hvad der er sket i turen. Hvis spilleren har vundet bliver spilleren sat til at have vundet. Til sidst bliver turen ændret til den anden spiller medmindre den nuværende spiller har fået en ekstra tur.

6.1 Minimumskrav og installationsguide

Spillet kan med sin meget simple grafiske brugerflade køre på størstedelen af moderne computersystemer. Nedenfor er en liste over hardware- og softwarespecifikationerne, som kan kører spillet 100%:

- **Hardware:**
  - Systemets interne hardware skal have en regnekraft sammenlignelig med computerne i DTU's Databarer.
  - Tastatur.
  - Skærm m. minimumsopløsning på 800x600.

  - Computermus.

- **Software:**
  - Styresystem:
    * Windows 8.1 eller nyere.
    * Mac OS X “El Capitan” vs. 10.11.6 eller nyere.
  - Softwarepakker:
    * Java 8.0 - https://www.java.com/en/download/

Spillet kan køre på både Windows og OSX, men er ikke blevet testet på en Linux-distro.

Installationsguide:

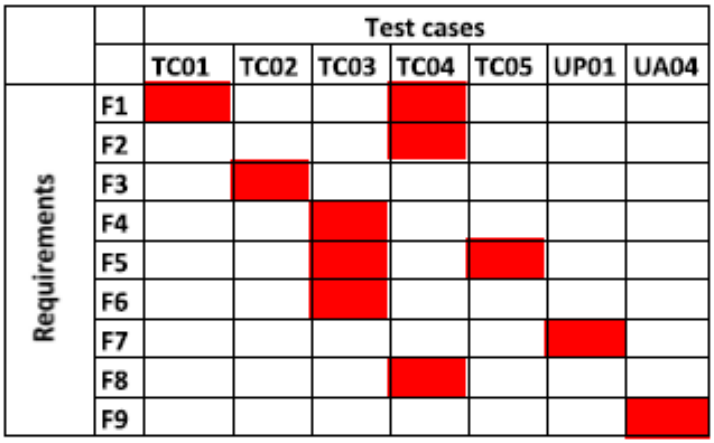
Åben mappen”33_CDIO2”, og dobbelt-klik på mappen der hedder ”Installation”. For Windows-brugere, dobbelt-klik på ”CDIO2_WIN.bat”. For OSX-brugere, dobbelt-klik på ”CDIO_2OSX.sh”.

Såfremt spillet ikke starter, kan du prøve følgende fejlløsning:

- Opdater/geninstaller din Java-software, og genstart dit system. Du skal bruge Java 8.0, for at kunne køre spillet.
- Hent spil-mappen igen fra CampusNet, og forsøg at køre spillet igen.
- Genstart din computer, og prøv igen.

7 Test

Efter at have udarbejdet Java-koden til spillet, så testes der om koden bearbejder inputs og outputs korrekt. Til at teste kodens forskellige metoder, anvendes Eclipses indbyggede testing framework JUnit. Disse tests bliver ikke uddybet i rapporten, medmindre de tester et specifikt krav.
Der opstilles en trace-ability matrix til at få et overblik over projektkravene, for at teste om kravene er opfyldt. Figur (11) viser traceability matricen for projektet, hvor de forskellige test cases står horizontalt (TC01, TC02..) og projektkravene står vertikalt (F1-F9).



Figur 11: Trace ability Matrix

Ud fra trace ability matricen ses det at alle projektkravene for projektet er testet igennem. Se nedenfor for en uddybende beskrivelse af de forskellige test cases:

- **Test case TC01** - Tester om spillet kan spilles af to spillere, og om deres spillernavne vises korrekt.
- **Test case TC02** - Tester om spilleren lander på feltnummerert som er identisk med spillerens terningesum.
- **Test case TC03** - Tester om felterne har den rigtige effekt på spillernes pengebeholdning. Der testes også om spillet viser det rigtige output, når spillerne lander på de forskellige felter.
- **Test case TC04** - Tester om der findes en vinder når en spiller opnår en pengebeholdning på 3000. Test casen tester derfor også om der skiftes tur imellem de to spillere.
- **Test case TC05** - Tester om The Werewall feltet, giver spilleren en ekstra tur.
- **Test case UP01** - Tester om spillerne starter med en pengebeholdning på 1000.
- **Test case UA04** - Tester om spillernes pengebeholdning kan blive negativ.

Alle testcasene er med til at sikre at produktet virker som det skal, og er funktionsdygtigt. Næsten alle testene kom tilbage positivt bekræftende, og produktet står derfor klar til en acceptance test, som senere vises til kunden.

7.1 Acceptance test

Acceptance testen bliver udført i samarbejde med kunden. Der gennemgås kort projektets forløb, med en forklaring om de basale funktioner af spillet. Man beder kunden gennemgå installationsguide, og godkende for kundens eget brug af spillet. Da kunden anvender en Windows-computer, vælger vedkommende at dobbeltklikke på ”CDIO2_WIN.bat”. Spillet startes, og man gennemkører et par spil for at vise at programmet virker.

Efterfølgende gennemgås en række småtests der er lavet undervejs, med reference til problemformuleringen og traceability matricen. Der dokumentereres at produktet overholder de krav som kunden selv har fremsat, samt en række krav som er opstillet efterhånden.

Kunden gennemgår selv de dokumenteret JUnit tests, og bekræfter resultaterne. Kunden godkender produktet.

8 Forbedringsforslag

I løbet af arbejdet med CDIO2, er man stødt på flere småting som man kan ændre på, for at forbedre brugeroplevelsen.

- **Ændringer i GUI'ens farvesammensætning:** GUI'ens nuværende farvetema står meget skarpt, med sin neongrønne farve. Et forbedringsforslag kunne f. eks. være en mere blød farve, eller endda mulighed for brugeren selv at vælge imellem flere forskellige farvetemaer.
- **Ændringerne i spillernes point tydeliggøres:** Med den nuværende GUI beregnes spillerens pointsum når spillerens tur afslutter - medmindre spilleren lander på Werewall-feltet. Det er lidt uoverskueligt præcis hvor meget feltets værdi påvirker spilleren point-sum.Evt tilføje Et lille felt i GUI'en med beregninger, kunne gøre spillerens forståelse for forløbet bedre.
- **Fjern konsollen - Ren GUI:** En anden måde som kunne forbedre programmet er ved helt at fjerne konsol-vinduet, og lave et felt i GUI'en, som kan håndtere keyboard-inputs. Ved at fjerne konsol-vinduet, eliminere forstyrrende elementer fra brugerfladen, og forbedrer brugeroplevelsen.
- **Felt-tallene passer ikke:** Der er i alt 11 forskellige felter, men spillerne "starter"på felt "2". Da to terninger maksimalt kan lave 11 forskellige værdisummer, er spillerne nødt til at starte på det første felt i række, "2", som man lander på igen ved at slå to ettere. Det er egentlig ikke muligt at forbedre dette, på grund af værdi-begrænsningerne der stammer fra terningerne. En løsning kunne være at oprette et 12. felt, og gøre det muligt for terningsummen at blive "1", og derved få 12 felter.
- **Spillet kan ikke genstartes igen efter endt spil:** Efter at spillet slutter, kan spillerne ikke genstarte spillet igen. En mulig løsning til dette kunne være at indsætte en knap, som når den bliver trykket på, genstarter Game-metoden igen, og starter spillet igen.

I det sidste CDIO projekt, blev der introduceret til Arrays i Java, en funktion som der kunne have draget nytte af i forbindelse med udviklingen af dette projekt. Dog, da man havde lavet produktet færdig uden brug af Arrays, blev der valgt ikke at introducere arrays til denne kode.

I løbet af dette CDIO projekt er der blevet præsenteret for Inheritance i Java, har man ikke brugt dette. Felterne på boardet er dog så lig hinanden, har man afgjort at det ikke kan svare sig at lave den arbejdsindsats. Nedarvning imellem felter kan dog blive relevant i forbindelse med en eventuel udvidelse af dette produkt.

9 Konklusion

I dette projekt skulle der programmeres et brætspil, som skulle kunne spilles af to spillere. Denne rapport dokumenterer, hvordan og hvorledes man har analyseret, designet, implementeret og testet kundens vision, således at brætspillet opfylder alle de krav kunden har sat. Derudover skal der også opfyldes de krav man har fundet nødvendige for at kunne bearbejde kundens vision på en fornuftig måde (programmeringsmæssigt). Overordnet set er man kommet frem til 10 klasser (udover den udleveret GUI.jar), som er blevet designet og implementeret med henblik på GRASP-koncepterne ”low-coupling”og ”high-cohesion”. Generalt set er samarbejde i og udførelsen af projektet gået godt og efter planen og man står nu med et færdig produkt som lever op til kundens vision.

10 Bilag

10.1 Formelle Test

Følgende er en pdf med alle de formelle test vi har foretaget. De er markeret med separate sidetal.

Test case ID	TC01
Summary	Tests if the entered player names are displayed correctly on the screen.
Requirements	-
Precondtions	The game starts.
Postconditions	The game displays the players names on the screen.
Test procedure	1. Player 1 enters his name. 2. Player 2 enters his name.
Test data	Player 1 name = "Abc".Player 2 name = "Efg".
Expected result	The correct value of the player's input.
Actual result	1. "Abc"2."Efg"
Status	Passed
Tested by	Jeppe Thougaard Nielsen
Date	Thursday, October 27, 2016
Test environme	Eclipse 4.6.1 on Windows 10 64-bit

Test case ID	TC02
Summary	Tests if the player lands on the field equal to the value of the dice roll.
Requirements	F3
Precondtions	The two players have entered their player names.
Postconditions	The player moved to the field equal to his dice roll.
Test procedure	The player rolls the dice.
Test data	The dice values are: 2 and 4
Expected result	The players field value is equal to the value of his dice roll.
Actual result	Field value = 6
Status	Passed
Tested by	Jeppe Thougaard Nielsen
Date	Thursday, October 27, 2016
Test environme	Eclipse 4.6.1 on Windows 10 64-bit

Test case ID	TC03
Summary	Tests if the visited field has the expected effect on the affected player.
Requirements	F5
Precondtions	The two players have entered their player names. The starting player has rolled the dice.
Postconditions	The players account correctly subtracts or adds the field's value.
Test procedure	The player rolls the dice. Player lands on field. The field has an effect on the players account. (Tested using the testUnfairDice Branch).
Test data	The dice values are: 1 and 1.
Expected result	The player is awarded 250 points, and his account is now 1250.
Actual result	Player 1's account = 1250.
Status	Passed
Tested by	Mads Stege
	Friday, October 28, 2016
Test environme	Eclipse 4.6.1 on Windows 10 64-bit

Test case ID	TC04
Summary	Tests if a winner is found when one of the players reach 3000 points.
Requirements	F8
Precondtions	The two players have played several turns, and one of the players are about to reach 3000 points.
Postconditions	The game found a winner and it is no longer possible to roll the dice.
Test procedure	1. Player 1 has 2750 points. 2. Player 1 rolls 2 (1+1). 3. Player 1 recieves 250 points from field 2 (Tower). 4. Player 1 has reached 3000 points.
Test data	Player 1 account balance = 2750.Player 1 rolled dice sum = 2.
Expected result	Player 1 to win the game as he reached 3000 points.
Actual result	Player 1 won the game.
Status	Passed
Tested by	Jeppe Thougaard Nielsen
Date	Thursday, October 27, 2016
Test environme	Eclipse 4.6.1 on Windows 10 64-bit

Test case ID	TC05
Summary	Tests if the field "The Werewall" correctly rewards the player with an extra turn.
Requirements	F5
Precondtions	The two players have entered their player names. The starting player has rolled the dice.
Postconditions	The player is moved to the field "The Werewall".
Test procedure	The player rolls the dice. Player lands on the field "The Werewall". The field has an effect on the players account, and afterwards gives the player another turn. (Tested using the testUnfairDice Branch).
Test data	The dice values are: 6 and 4.
Expected result	The players account is subtracted 80 point, and the player receives another turn.
Actual result	Player 1's account = 920 and Player 1 has been given another turn.
Status	Passed
Tested by	Mads Stege
	Friday, October 28, 2016
Test environme	Eclipse 4.6.1 on Windows 10 64-bit