CDIO 2

Gruppe nr: 45

Denne rapport indeholder 20 sider inkl. bilag og denne side.

Navn: Mads Martin Dickmeiss Hemer

S-nummer: s170185

Email: s170185@student.dtu.dk

Navn: Malte Brink Kristensen

S-nummer: s185039

Email: s170185@student.dtu.dk

Navn: Martin Dall

S-nummer: s163728

Email: s163728@student.dtu.dk

Navn: Nicolai de Thurah Wulff

S-nummer: s185036

Email: s185036@student.dtu.dk

Navn: Neal Patrick Norman

S-nummer: s060527

Email: s060527@student.dtu.dk

Navn: Camilla Bruun Simonsen

S-nummer: s185038

Email: s185038@student.dtu.dk

Indholdsfortegnelse

Indledning

Det følgende er et spil, vi har udviklet til IOOuterActive. Det er et simpelt spil, der i dets nuværende form kan betragtes som en slags alpha-version. Spillet spilles af to spillere, der på skift slår med 2 terninger og lander på et felt med numrene fra 2 - 12. Der udskrives en kort tekst om det felt, hver spiller lander på, og spilleren vinder eller taber et pengebeløb. Hver spiller begynder spillet med en pengebeholdning på 1000 og vinder spillet ved at nå 3000.

Analyse

Krav

I de følgende afsnit specificerer vi kravene i en vision, en kravspecifikation, en navneordsanalyse og en domænemodel.

Vision og kravspecifikation

Det følgende er baseret på kundens egen formulering af deres vision. Vores vision er at udvikle et spil mellem to personer, der kan spilles på maskinerne i DTU's databarer. Spillet går i korte træk ud på, at spillerne på skift slår med to terninger og lander på et felt fra 2-12 (1 er startfelt). Når en spiller landet på et felt, har det en positiv eller negativ effekt på deres pengebeholdning. Spillerne starter med en pengebeholdning på 1000, og spillet er slut, når en spiller når en pengebeholdning på 3000.

Kunden har endvidere udleveret en liste over de felter, der skal indgå i spillet, hvor hvert felt har angivet et navn og en effekt. Kunden har også specificeret en række ikke-funktionelle, supplerende kravspecifikationer, som rapporten senere vil komme ind på (se afsnit ??).

Ud fra kundens visionsformulering har vi tolket os til en række krav, som vi i det følgende vil opstille. Vi har dog bemærket, at der mangler nogle detaljer i deres beskrivelse af spillet, som vi derfor selv tage stilling til, og som også vil medføre krav til spillet. Dette kan f.eks. være: Hvad sker der, hvis en spiller får en negativ pengebeholdning? Her har vi valgt, at det ikke skal være muligt at have negativ pengebeholdning - altså at en spillers pengebeholdning forbliver på 0, indtil spilleren lander på et felt med positiv effekt.

Vi vil desuden introducere nogle krav til spillet, der har til formål at øge spillernes fornemmelse af rent faktisk at spille et spil. I spillet, som det er beskrevet i visionen, har spilleren som sådan kun ét valg, nemlig at slå med terningerne (eller ej). Vi forestiller os dog, at spillerne først egentlig oplever, at de spiller et spil, hvis de præsenteres for flere valg, som de aktivt må tage stilling til. For at opnå dette vil vi blot sørge for, at spilleren i sin tur præsenteres for en menu

med med nogle få valg, såsom at “slå terningerne”, “giv op” eller “se stillingen”, på trods af at de to sidstnævnte valg som sådan ikke er særligt meningsfulde.

Funktionelle krav til systemet:

K1	Spillet skal kunne spilles af præcis to personer.
K2	Spillerne skal på skift kunne slå med to terninger, der er tilfredsstillende symmetriske.
K3	Spillerne skal kunne lande på forskellige felter afhængig af terningeslaget (se tabel 1 for specifikation)
K4	Spillerne skal have en pengebeholdning, der starter på 1000.
K5	Spillernes pengebeholdning skal kunne hhv. øges og mindskes ved at lande på felter (se tabel 1 for specifikation).
K6	Spillernes pengebeholdning må ikke kunne blive negativ.
K7	At lande på et felt skal udskrive en tekst omhandlende det aktuelle felt.
K8	Spillet skal slutte, når en spiller opnår en pengeholdning på 3000 eller over.
K9	Spillet skal kunne spilles på forskellige sprog (UI skal kunne vise tekst på forskellige sprog, og spillerne skal kunne skifte imellem disse).
K10	Spillet skal ved hver tur præsentere en menu med følgende valgmuligheder: - Slå terninger - Giv op - Se stilling
K11	Spillet skal starte med visningen af en hovedmenu med følgende valgmuligheder: - Starte et spil - Læse reglerne - Skifte sprog - Afslutte programmet

Spillets felter

1.	(Man kan ikke slå 1 med to terninger)	
2.	Tower	+250
3.	Crater	-100
4.	Palace gates	+100
5.	Cold Desert	-20
6.	Walled city	+180
7.	Monastery	0
8.	Black cave	-70
9.	Huts in the mountain	+60
10.	The Werewall (werewolf-wall)	-80, men spilleren får en ekstra tur.
11.	The pit	-50
12.	Goldmine	+650

Tabel 1: Oversigt over felter, deres tilhørende terningeslag samt likviditetsvirkning.

Ud over de funktionelle krav er der desuden en række supplerende specifikationer til projektet. Dette inkluderer ikke-funktionelle krav, useability-krav og krav, der relaterer sig til selve udviklingsarbejdet.

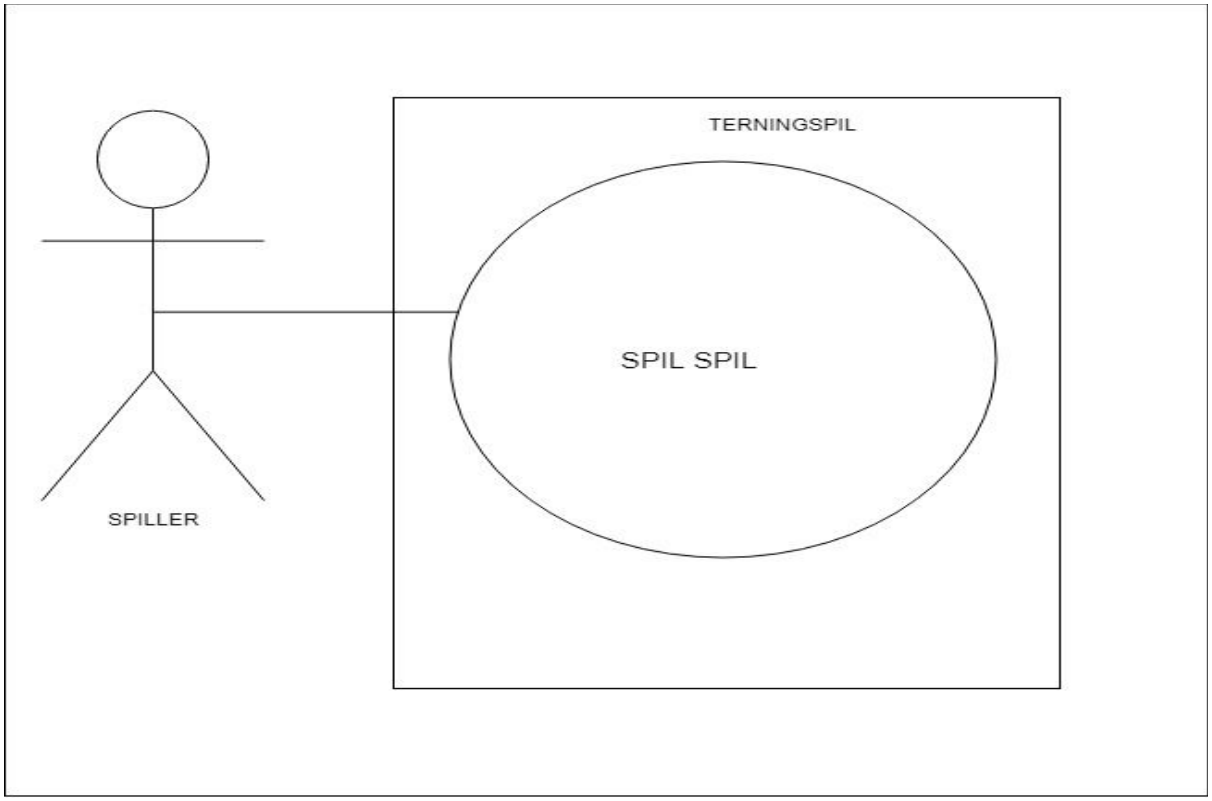
Supplerende kravspecifikationer

- Kunden skal kunne se hver enkelt gruppemedlems bidrag.
- Kunden skal kunne se hvordan udviklingen er foregået
- Der skal leveres et git-repository så kunden kan se og følge med i commits
- GRASP-principperne skal overholdes
- Det skal være let at skifte til andre terninger
- Spillet skal kunne spilles på DTU´s databar, uden bemærkelsesværdige forsinkelser

Use case diagram

Vores ene use case (UC1) beskriver den ene funktion, som vores system indeholder – nemlig et spil, der spilles:

UC1



Figur 1: Use case 1

Use case beskrivelse

Nedenfor har vi beskrevet vores use case. Først brief, så casual, så fully dressed. Ud fra use casen laver vi en navneordsanalyse, en domænemodel og et systemsekvensdiagram.

UC1

Brief:
Spillerne starter spillet og skiftes til at slå med to terninger. Afhængig af øjnenes værdi lander spiller på forskellige felter, der påvirker deres pengebeholdning og spillets gang. Den spiller der først får 3000 på kontoen vinder.

Casual:
Hovedsuccesscenarie:
Spillerne starter spillet og skiftes til at slå med to terninger. Afhængig af øjnenes værdi lander spiller på forskellige felter, der påvirker deres pengebeholdning og spillets gang. Den spiller der først får 3000 på kontoen vinder.

Alternative scenarier:

- En spiller giver op. Spillet slutter straks.
- En spiller vil se stillingen. Stillingen vises

Fully dressed:

Scope: Terningespil.
Level: User goal.
Primær aktør: Spiller.
Interessenter:

- Spiller: Vil gerne spille (og vinde) et spil med en modspiller.
- IOOuterActive: Vil gerne have tilfredse spillere.

Postconditions: Et spil er afsluttet og en vinder er fundet.
Hovedsuccesscenarie:

1. Spillerne præsenteres for en hovedmenu, hvor de kan vælge enten at starte et spil, læse reglerne, skifte sprog eller afslutte programmet.
2. Spillerne starter et spil.
3. Spillet vælger automatisk og tilfældigt hvilken af de to spillere, der får lov at starte.
4. Spilleren, der starter (spiller 1), præsenteres for en menu med de valg, han/hun har.
5. Spiller 1 vælger at slå med terningerne.
6. Summen af terningerne findes, og spiller 1 lander på det felt, der har har det nummer, som summen angiver.
7. Spillet udskriver en tekst om det felt, som spiller 1 er landet på.
8. Spiller 1 opnår den effekt, som feltet har, hvilket enten kan være en positiv eller negativ effekt på spillerens pengebeholdning, eller en ekstra tur.
9. Turen går nu til spiller 2, som gennemgår det samme flow som spiller 1 netop har været igennem (dvs. punkt 3-7).
10. Spillet forsætter sin gang som beskrevet ovenfår, hvor de to spillere får en tur på skift.
11. Når en spiller har 3000 eller flere point, slutter spillet, og der udskrives en besked om, hvilken spiller, der har vundet, samt at spillet nu er slut.
12. Spillerne præsenteres igen for hovedmenuen, hvor de har samme valg som i punkt 1.
13. Når spillerne ikke ønsker at spille længere, lukker de programmet enten ved at vælge “luk programmet” i hovedmenuen, eller på anden vis.

Alternative scenarier:

1. Spiller beder om at læse spillets regler.
    a. Systemet viser reglerne.
2. Spiller beder om at stoppe med at spille
    a. Systemet lukker ned.

Forekomstfrekvens: Konstant.

Navneordsanalyse

Spiller, spil, felter, terninger, penge, konto, pengebeholdning, sprog.

Vi vil udvikle spillet, så det relativt let kan tilføres en GUI; vi tænker altså også på et begyndende plan i grafisk brugergrænseflade. Derfor introducerer vi yderligere navneordet "spilleplade", som egentlig består af de 11 felter, man kan lande på i spillet.

Domænemodel

Domænemodellen viser, at der til hver spiller er et spil, og at der til hvert spil er to spillere; at der til hver spiller er to terninger, og at der til hvert sæt terninger er en spiller. Derudover...

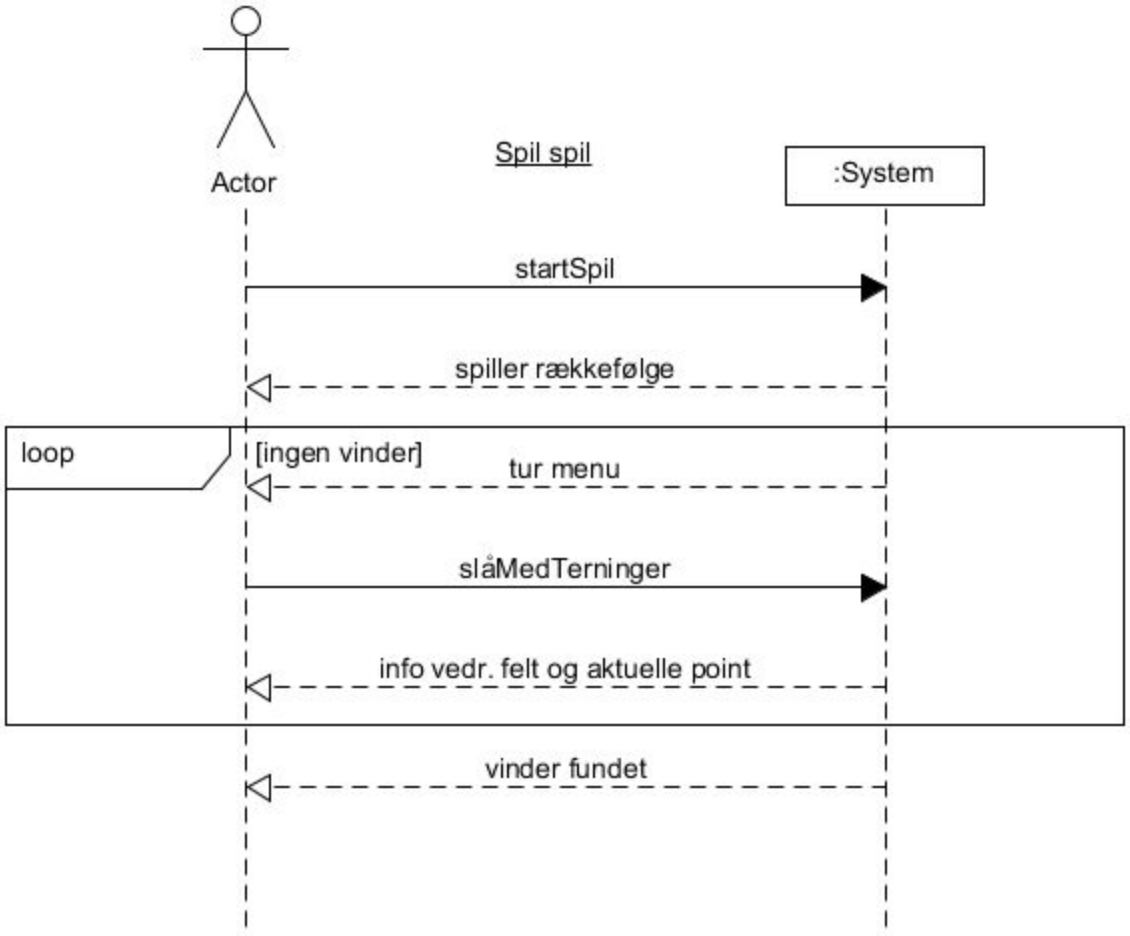
Nye konceptuelle klasser ift. CDIO 1:

Pengebeholdning

Spilleplade
Felter på spillepladen

Systemsekvensdiagram

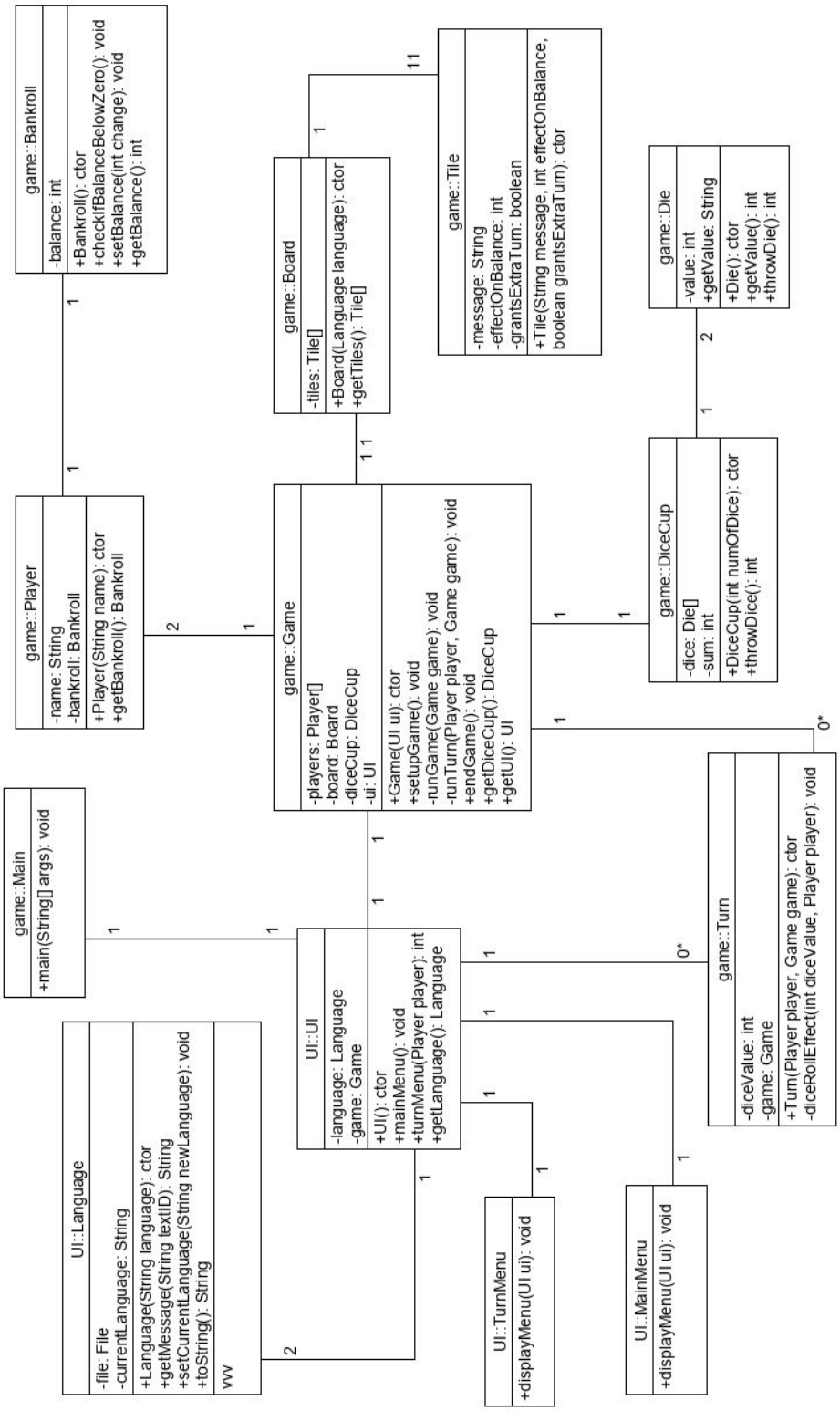
Følgende systemsekvensdiagram beskriver Use case 1. Spilleren beder om at spille et spil, hvorefter systemet informerer om hvilken spiller, der har turen, samt hvilke valgmuligheder spilleren har i form af en menu. Spilleren vælger selvfølgelig at tage sin tur og kaste med terningerne (fordi de er i gang med at spille et spil). Systemet informerer herefter om effekten af terningslaget. Næste spiller præsenteres herefter for menuen og kaster med terningerne, så længe der ikke er fundet en vinder osv. Med andre ord køres en løkke indtil en af spillerne har vundet.



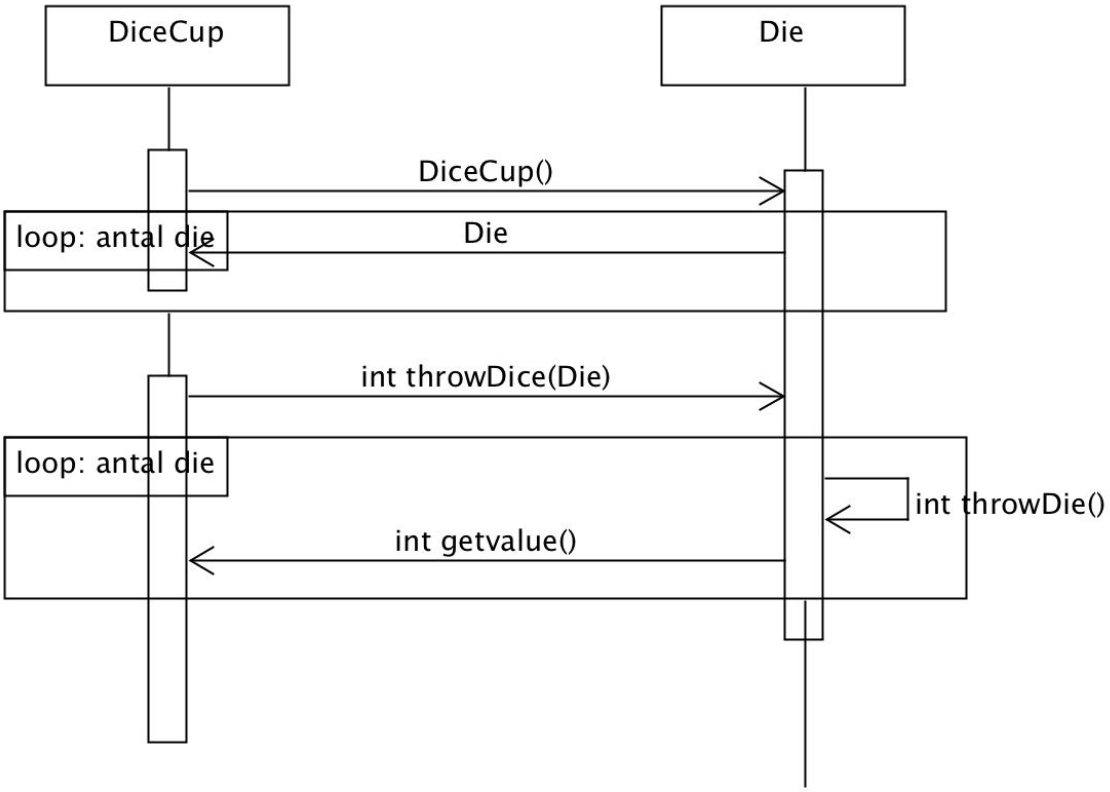
Design

På næste side starter vi med vores designklassediagram, da det ikke kan være på denne side.

Designklassediagram



Sekvensdiagram



Sekvensdiagrammet viser, DiceCup instantierer et antal Die. DiceCup vælger en die og kører metoden throwDice, som kører det antal gange, der er terninger. DiceCup.throwDice kalder getvalue fra Die og summerer værdierne. Vi er opmærksomme på at kun en af vores skabte die bliver givet videre og den bliver brugt flere gange til at rulle, men vi har valgt at kode det sådan for at leve op til de krav, kunden har specificeret.

GRASP

I dette afsnit beskrives vores forsøg på - med varierende grader af succes - at anvende GRASP-designprincipperne.

De to klasser, UI og Game, er tiltænkt at være controllere i henholdsvis UI- og game-pakken. Dette er blandt andet realiseret ved at lade dem styre spillets gang for hver deres ansvarsområde. De andre klasser i de to pakker er således tiltænkt at skulle støtte UI og Game i afviklingen af spillet. Selve spillet (programmet) startes ved instantieringen af et UI-objekt, der

samtidig instantierer et Game-objekt. Både Game og UI fungerer samtidig som de primære creators i pakkerne (og nok også information experts).

Umiddelbart har vi dog muligvis lagt for meget af spillets funktionalitet og logik i selve UI og Game-klasserne, og har dermed brudt de to designmønstre (eng.: design patterns) høj sammenhængskraft (eng.: high cohesion) og lav sammenkobling (eng.: low coupling). Et eksempel på dette er, at de bruger-inputs, der modtages via en scanner, alle er placeret i selve UI-klassen. De kunne måske med fordel være flyttet ud i de klasser, der indeholder de forskellige menuer. Således kunne et UI-objekt for eksempel bare kalde en metode, der laver/viser en ny hovedmenu og bare modtager en returværdi, der nu er relevant ift. brugerens valg.

I de klasser, der ikke er UI eller Game, har vi forsøgt at opfylde lav sammenkobling og høj sammenhængskraft, ved at lave fokuserede og simple metoder og ved at hente information hos information experts i stedet for at have den direkte tilgængelig.

Implementering

Følgende funktion (runGame) kan betragtes som hjertet i spillet. Den starter en løkke, der bliver ved med, via runTurn metoden, at køre ture indtil en vinder er fundet. Hvis en spiller, efter afviklingen af en tur, har opnået en score på 3000, stopper løkken - ellers ændres den nuværende spiller og en ny tur køres med en ny spiller.

```
private void runGame(Game game) {
   boolean winnerFound = false;
   int currentPlayerArrayIndex = 0; //players[0] always goes first.
   do {
       runTurn(players[currentPlayerArrayIndex], this);

       if (players[currentPlayerArrayIndex].getBankroll().getBalance() >= 3000) {
           winnerFound = true;
           game.endCurrentGame(players[currentPlayerArrayIndex]);
       } else {
           switch (currentPlayerArrayIndex) {
               case 0:
                   currentPlayerArrayIndex = 1;
                   break;
               case 1:
                   currentPlayerArrayIndex = 0;
                   break;
           }
       }
   } while (!winnerFound);
}
```

Klassen bankroll i gamepakken styrer spillernes pengebeholdninger. Den indeholder fire funktioner: Hver spiller starter med at have en beholdning på 1000, beholdningen kan ikke blive negativ, funktionen til at justere beholdningerne efter hver spillers tur og en get-funktion, der kan kaldes fra UI-klassen. Det var et krav fra opgaven fra IOOuterActive, at bankroll var en klasse for sig - ellers kunne funktionerne måske have været lagt i andre klasser. Idet spillet skal være let at arbejde videre på i forskellige retninger, og vi selvfølgelig fortsat arbejder efter GRASP-principperne høj sammenhængskraft og lav sammenkobling, giver det desuden mening at fordele spillets funktioner på mange forskellige klasser. Bankroll-klassen må primært betragtes som en information expert.

```
package game;

public class Bankroll {

   private int balance;

   //The game starts with each player having 1000
   public Bankroll() {
       this.balance = 1000;
   }


   //The bankroll can't be negative
   private void adjustIfBalanceBelowZero(){
       if(this.balance < 0 ){
           this.balance = 0;
       }
   }

   public void adjustBalance(int change){
       this.balance += change;
       adjustIfBalanceBelowZero();
   }

   //Get balance
   public int getBalance() {
           return this.balance;
   }

}
```

Følgende metoder fra Language klassen er den måde, hvormed programmet finder beskeder i en tekstfil, således at de kan printes til konsollen. Denne metode er således også den måde, hvorpå muligheden for nemt at kunne skifte og tilføje nye sprog realiseres. Filen, som metodens scanner, læser fra skal struktureres således at en linje indeholder et ID (for eksempel main_menu_header) efterfulgt af en linje med beskeden. Metoden kører igennem tekstfilen linje for linje indtil den finder ID'et og returnerer derefter den næste linje

```
public String getMessage(String textID) {
   String message = "";
   try {
       Scanner scanner = new Scanner(file);
       while (scanner.hasNextLine()) {
           String line = scanner.nextLine();
           if(line.equals(textID)) {
               message = scanner.nextLine();
               break;
           }
       }
       scanner.close();
   } catch(FileNotFoundException e) {
       System.out.println("text not found in language file!");
   }
   return message;
}
```

Test

Det ville være ønskværdigt at foretage integrationstest.

J Unit

Vi når kun i denne omgang at lave en test af, om pengebeholdningen ikke kan blive negativ. Testen sætter beholdningen til at blive negativ, og undersøger derefter, om spillet automatisk sætter pengebeholdningen til nul.

Konfiguration

Programmet leveres som en mappe, der indeholder en batch-fil (.bat). Programmet kan køres via denne fil, dog kræver det en computer med Java Runtime Environment (JRE) installeret og et windows-styresystem.

Vi afleverer et GIT repository (mappen .git) hvor det kan ses hvilke commits, der er foretaget, og af hvem. Plus en zipfil med IntelliJ-projektet (i zip-filen af projektmappen bliver .git-mappen inkluderet automatisk).

Konklusion

Spillets i dets nuværende form kan betragtes som en slags alpha-version. En stor del af den primære funktionalitet fungerer, men en hel del sekundære funktioner virker dårligt eller slet ikke. Et spil kan godt køres, men det er ikke sikkert, at det slutter ordentlig eller i det hele taget kører uden bugs. Vores unit-tests viser dog, at flere af programmets funktioner fungerer udmærket for sig selv. Fremtidigt arbejde kunne hjælpes på vej af mere grundigt designarbejde, så det eksempelvis vil blive nemmere at kunne overholde GRASP-principperne gennem større overblik over programmets struktur.

Referencer

Larman, C. (2004) *Applying UML and patterns: an introduction to object-oriented analysis and iterative development.* Upper Saddle River: Pearson Education.

Lewis & Loftus (2011) *Java Software Solutions Foundations of Program Design*, Pearson Education.

Bilag

Felter og deres tekster

Felter og deres tekster på engelsk

1. –

2. Tower
   ”Yay, you found 250 in a tower. Getting richer!”

3. Crater
   ”Oh no! You fell into a crater. -100.”

4. Palace gates
   ”You entered the palace gates and received 100! Good for you.”

5. Cold Desert
   “Uh, you landed in the cold desert. -20.”

6. Walled city
   ”You entered the walled city. Nice! You receive 180”

7. Monastery
   ”You are passing some time in the monastery away from the trouble of the rest of the world. No effect on your economy.”

8. Black cave
   ”Oh no. You landed in the black cave. The cave monsters took 70.”

9. Huts in the mountain
   ”You found 60 in the huts in the mountain, hooray.”

10. The Werewall (werewolf-wall)
    ”Oops, you touched the Werewall and the werewolf ate 80. You get to cast the dice again though.”

11. The pit
    ”You fell into a pit and lost 50”

12. Goldmine
    ”You found gold in the mountains and sell it for 650, you’re rich!”

Felter og deres tekster på dansk

1. –

2. Tårn
   ”Yay, du fandt 250 I et tårn. Du bliver rigere!”

3. Krater
   ”Åh nej! Du faldt ned i et krater. -100.”

4. Paladsets porte
   ”Du passerede paladsets porte og fik 100! Godt for dig.”

5. Den kolde ørken
   ”Uh, du landede i den kolde ørken. -20.”

6. Den indhegnede by
   ”Du er gået ind i den indhegnede by. Nice! Du får 180.”

7. Klosteret
   ”Du fordriver tiden lidt i klosteret, væk fra resten af verdens problemer. Ingen effekt på din økonomi.”

8. Sort grotte
   “Åh nej. Du landede i den sorte grotte. Grottemonstrene tog 70 fra dig.”

9. Hytter i bjerget
   ”Du fandt 60 i hytterne på bjerget, hurra.”

10. Varvæggen (varulvevæg)
    ”Ups, du rørte ved Varvæggen og varulven åd 80. Du får dog lov at kaste terningerne igen.”

11. Skakten
    ”Du faldt ned i en skakt og mistede 50”

12. Guldmine
    “Du har fundet guld i bjergene og sælger det for 650, du er rig!”

Spilleregler

Spilleregler dansk

Spillerne slår på skift med 2 terninger og lander på et felt med numrene fra 2 - 12. At lande på hvert af disse felter har en positiv eller negativ effekt på spillernes pengebeholdning. Der udskrives en kort tekst om det aktuelle felt. Spillerne begynder spillet med en pengebeholdning på 1000. En spiller vinder spillet ved at nå 3000.

Spilleregler engelsk

The players take turns rolling two dice, landing on tiles numbered 2-12. Landing on each of these tiles have a positive or negative effect on the players’ bankroll. A short text about the

relevant tile is printed. The players have a bankroll of 1000 when the game starts. A player wins the game by reaching 3000.