Rapport - CDIO del 2

Må ikke anvendes i undervisningen

Afleveret 10/11-2017

Udarbejdet af

S175219 - Thomas Løvendahl Vestergaard

S175216 - Kasper Bøgvad Nielsen

S175213 - Casper Kyster Andersen

S172139 - James Giles

S144260 - Tahany Nemer

S160435 - Cornelius Frost Schilling Hedegaard

S163157 - Abdirisaq Abdiqadir Mohamud Farah

Timeregnskab

Dato	Deltager	Design	Impl.	Test	Dok.	Andet	Ialt
29-09-2017	Thomas	1	6	1	3	1	12
29-09-2017	Kasper	3	5	1	4	1	14
29-09-2017	Tahany	4	10	2	2	1	19
29-09-2017	James	0	0	0	0	0	0
29-09-2017	Casper	1	6	2	5	1	15
29-09-2017	Cornelius	4	8	5	2	1	20
29-09-2017	Abdirisaq	0	0	0	0	0	0

Indholdsfortegnelse

1 Indledning

På baggrund af spillet "Terning", er der fra IOOuterActive blevet opstillet endnu en opgave omkring et nyt spil. Det ønskede spil er et brætspil, som skal kunne spilles mellem 2 personer.
Det skal foregå således, at spillerne skiftevis kaster med 2 terninger, lander på et felt der justerer spillerens "guld", og derefter skifter tur til den efterfølgende spiller.
Dermed kan vi inkludere "Terning" i det nye spil, men den ændring at terningerne let skal kunne udskiftes til andre-sidede terninger.
Det er desuden et krav at det skal være muligt at ændre sproget af spillet, samt tilføje yderligere sprog på en nem måde, selvom at en oversætter ikke har programmeringserfaring med Java eller lignende programmeringssprog.

2 Analyse

Kravliste

Funktionelle krav:

K1. Spillers balance må aldrig gå i negativ.

K2. Spillerne skal miste og få penge ud fra hvilket felt de lander på.

K3. Spillerne skal skiftes om at slå med to seks-sidede terninger.

K4. Spillepladen skal have 11 forskellige felter, med forskellige funktioner og beskrivelser.

K5. Spillerne skal starte med en balance på 1000.

K6. Spilleren som når 3000 eller over i deres balance først vinder spillet.

K7. Spillet skal spilles mellem 2 personer.

K8. Efter hvert slag skal spilleren tilbage til start, således at en spiller ikke går fra felt 5 til felt 9 ved at slå 4.

Ikke funktionelle krav:

K9. Spillet skal være på engelsk, og være let at oversætte.

K10. Det skal være nemt at skifte terning i koden.

K11. Programmet skal kunne køres på DTU's databarer.

K12. Spiller og Konto klassen skal nemt kunne bruges i andre programmer.

K13. Forsinkelser skal højst være på 333 ms.

Krav i FURPS+

FURPS står for:

Functionality:

Spillerne skal starte med en balance på 1000.

Spillepladen skal have 11 forskellige felter, med forskellige funktioner og beskrivelser.

Efter hvert slag skal spilleren tilbage til start, således at en spiller ikke går fra felt 5 til felt 9 ved at slå 4.

Spilleren som når 3000 eller over i deres balance først vinder spillet.

Spillerne skal få og miste guld ud fra hvilket felt de lander på.

Spillerne skal skiftes om at slå med to seks-sidede terninger.

Spillers balance må aldrig gå i negativ.

Usability:

Spillet skal være på engelsk, og være let at oversætte.

Spillet skal spilles mellem 2 personer.

Reliability:

Spillet skal ikke udprinte nogle fejl under kørsel af det.

Performance:

Forsinkelser skal højst være på 333 ms.

Supportability:

Programmet skal kunne køres på DTU's databarer.

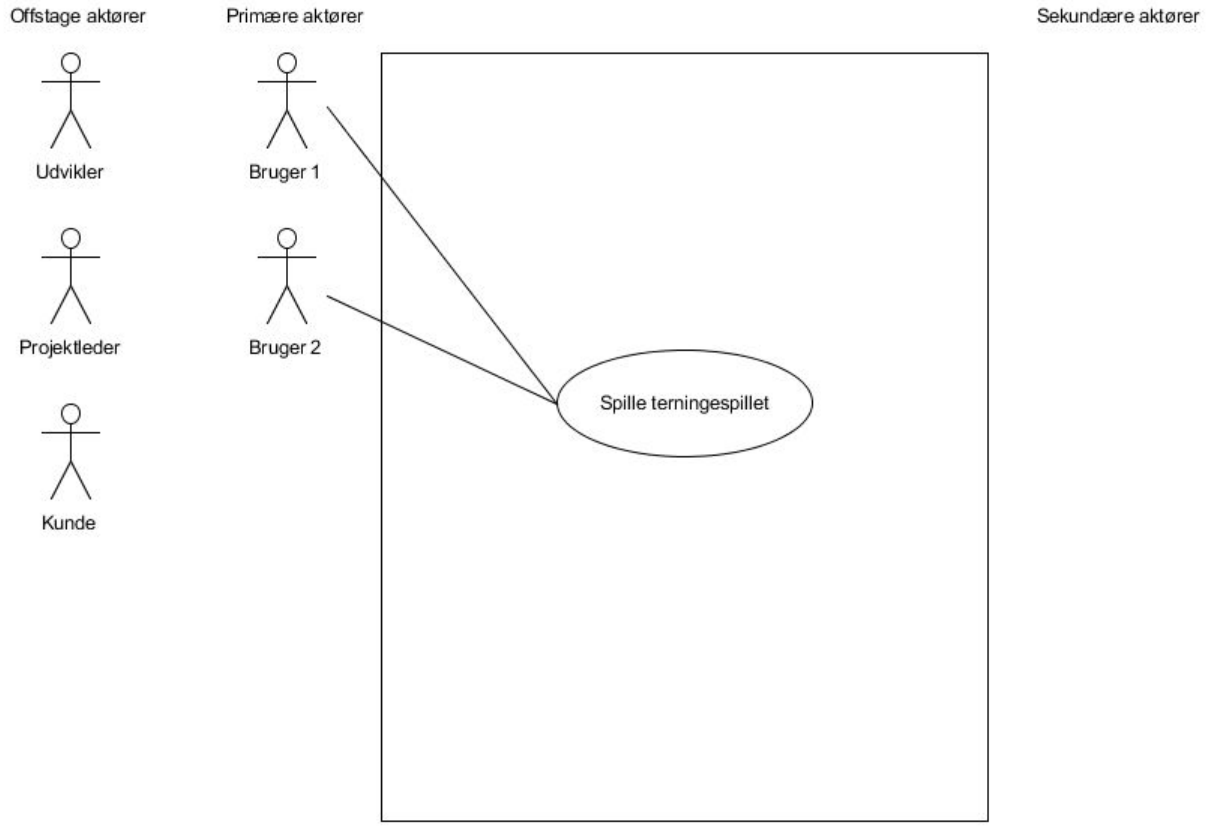
Det skal være nemt at skifte terning i koden.

Spiller og Konto klassen skal nemt kunne bruges i andre programmer.

Aktør- og interessentanalyse

Da vores spil er et "offline" spil, betyder dette at vi ikke har nogle aktører uden for brugeren af spillet, dvs. spilleren.
Der er derfor ikke nogle sekundære aktører, da der ikke er nogen mulighed for at interagere med spillet, uden for den lokale instans af det, d.v.s. alle spillets interaktioner er gennem brugeren.
Vi har dog off-stage aktørerne, som dog ikke er reelle aktører, men mere en form for interessent i spillet, da deres interesse er mere fokuseret på funktionaliteten af spillet. Dette involverer normalt virksomhedens bestyrelse (eller lignende) i et projekt som dette.

Use case diagram



Brief, Successscenarie:

To spillere tænder for spillet, og vælger et sprog samt deres brugernavne.
Derefter slår spillerne på skift med to terninger, hvorefter spilleren lander på et felt baseret på antallet af øjne slået og derefter tildeles/fratages guld baseret på det derpå landede felt.
Spilleren får på samme tid en besked om, hvilket felt spilleren er landet på, og hvor meget guld der tildeles/fratages.
Den første spiller der først får 3000 guld vinder spillet, og får en besked om dette af systemet, hvorefter terningerne ikke længere kan slås og spillet afsluttes.

Alternativt Scenarie:

Hvis en spiller slår 7 med terninger, hvor der hverken tildeles eller fratages guld fra spilleren, men turen går videre til den anden spiller.

Alternativt Scenarie:

Hvis en spiller slår 10 med terningerne, hvor spilleren mister 80 guld, men får et ekstra slag.

Alternativt Scenarie:

Hvis en spillers guld går i negativ, sættes guldbalancen med det samme til 0.

Øjne	Felt	Guld
2	Tårnet/Tower	+250
3	Krateret/Crater	-100
4	Paladsportene/Palads gates	+100
5	Den Kolde Ørken/ Cold Desert	-20
6	Den Indemurede By/ Walled city	+180
7	Klosteret/Monastery	0
8	Den Sorte Grotte/Black cave	-70
9	Bjerghytterne/Huts in the mountains	+60
10	Varulvevæggen/The Werewall	-80
11	Hullet/The pit	-50
12	Guldminen/Guldminen	+650

Fully Dressed:

Navn:

Spille Terningespil

Anvendelsesområde:

”Usikker – Spørg en underviser”

Primære Aktør:

Bruger 1 og Bruger 2

Interessenter mm.

Kunde: I dette tilfælde en repræsentant fra IOOuterActive (DTU). Kunden vil gerne have et produkt, der gør hvad der er forventet af det, da kunden betaler for at få lavet produktet.

Udvikler og Projektleder: Ønsker begge at blive betalt for at have lavet produktet, hvilket er muligt, hvis kunden er tilfreds med det færdige produkt.

Forudsætning:

Computeren har installeret java 1.8

Postcondition:

En spiller er erklæret vinder, og spillet er enten blevet afsluttet eller genstartet.

Succes Scenarie:

1. Spillerne tænder for spillet.
2. Spillet sætter sproget efter computerens sprog.
3. Spiller 1 indtaster et brugernavn.
4. Spiller 1 modtager 1000 guld.
5. Spiller 2 indtaster et brugernavn.
6. Spiller 2 modtager 1000 guld.
7. Spiller 1 trykker enter for at kaste 2 terninger.

8. Spiller 1 lander på et felt baseret på antallet af øjne.

9. Spiller 1 modtager eller mister guld baseret på det landende felt. (Se tabel 1)

10. Spiller 1 får en udskrift fra systemet om hvilket felt spilleren er landet på, og hvor meget guld spilleren har modtaget eller mistet, samt sin nye balance.

11. Spiller 2 trykker enter for at kaste 2 terninger.

12. Spiller 2 lander på et felt baseret på antallet af øjne.

13. Spiller 2 modtager eller mister guld baseret på det derpå landede felt. (Se tabel 1)

14. Spiller 2 får en udskrift fra systemet om hvilket felt spilleren er landet på, og hvor meget guld spilleren har modtaget eller mistet, og hvad hans balance er på.

[Systemet gentager trin 6 til 13, indtil en spiller har 3000 eller mere guld.]

15. Spillet lukker for inputs til at kaste med terningerne.

16. Spillet udskriver en besked, der siger at en spiller med 3000 eller mere guld har vundet.

17. Spiller giver mulighed for at afslutte spillet, eller genstarte spillet fra trin 2.

Alternative Scenarier:

A: I trin 6 el. 10, hvis en spiller slår ”7” med terningerne.

1. Spilleren lander på feltet ”Klosteret”.

2. Spilleren hverken modtager eller mister guld.

3. Spilleren får en udskrift fra systemet om at spilleren er landet på ”Klosteret”, og at spilleren hverken har modtaget eller mistet noget guld.

4. Turen gives videre til den anden spiller og der fortsættes som normalt.

B: I trin 6 el. 10, hvis en spiller slår ”10” med terningerne.

1. Spilleren lander på feltet ”Varulvevæggen”.

2. Spilleren mister 80 guld.

3. Spilleren får en udskrift fra systemet om at spilleren er landet på ”Varulvevæggen”, og at spilleren har mistet 80 guld, men at spilleren samtidig får et ekstra kast.

4. Turen gives ikke videre til den anden spiller, og den spiller der slog 10, kan i stedet kaste med terningerne igen.

C: Når en spiller i trin 8 el. 12 mister guld så spillerens guldbalance går i negativ.

1. Systemet sætter guldbalancen fra den negative værdi til 0.

2. Spilleren får en udskrift fra systemet om hvilket felt spilleren er landet på, og hvor meget guld spilleren har modtaget eller mistet. Derudover tilføjes der til udskriften at spilleren ikke kan gå i negativ, og derfor har fået sat sit guld til 0.

3. Turen gives videre til den anden spiller og der fortsættes som normalt.

D: På et hvilket som helst tidspunkt en spiller prøver at lukke spillet.

1. Spillet udskriver en besked, der spørger om spilleren ønsker at afslutte ved at indtaste ”exit”.

2. Spilleren indtaster ”exit”.

2a. Spilleren indtaster noget andet end ”exit”

    1. Spillet fortsættes.

3. Spillet afsluttes.

Specielle Ikke-Funktionelle Krav:

- Det skal være nemt at skifte terning i koden.

- Spiller og Konto klassen skal nemt kunne bruges i andre programmer.

Åbne problemer:

- Hvad hvis ingen af spillerne når 3000 guld?

Vision:

- **Beskrivelse af området/domænet**

Projektet omhandler et spil, der kan spilles mellem 2 personer på databarene på DTU. Spillet skal kunne slå 2 terninger, og placerer en spiller på et felt fra en liste af 11 felter baseret på summen af terningerne.
Hver spiller har en konto med penge, hvor hvert felt ændrer på kontoens guldbalance.
Derefter gives turen videre til den anden spiller, og der spilles indtil en spiller har nået et vis guldbeløb på deres konto, hvorefter den spiller vinder.

- **Beskrivelse af interessenter, deres mål og problemer**

Interessenter er personer, der ikke interagerer med det færdige produkt, men stadig er interesseret i det. Til dette projekt vil dette inkluderer kunde, udvikler og projektleder. Alle 3 er interessenter af monetære grunde.
Kunden ønsker ikke at betale for noget halvfærdigt, der ikke kan hvad det skal, hvor udvikler og projektleder ønsker en betaling for det udførte arbejde.
Brugeren vil også være en interessant samt en aktør til projektet, da det er brugeren der skal bruge spillet til underholdningsbrug.

- **Oversigt over systemets egenskaber**

Spillet giver brugeren mulighed for at indtaste et brugernavn, vælge mellem sprog og kaste med 2 terninger. Derudover viser spillet automatisk brugerens score efter hvert slag, samt hvilket felt brugeren landede på. Scoren ændrer spillet på, baseret på hvilket felt brugeren lander på, hvor en billedlig tekst fortæller om brugerens omgivelser.

- **Fordele ved brug af systemet**

Spillet kan have fordelen ved at fungere som afkobling i pauserne til forelæsningerne på DTU, eller blot som underholdning i mellem 2 studerende, hvilket kan lede til samtale og evt. et bedre studiemiljø..

- **Økonomi**

Bortset fra pengene i mellem kunde og projektleder/udvikler i forhold til udvikling af spillet, er der ikke brug for meget i forhold til vedligeholdelse, og det eneste der skal bruge penge på er at få spillet på computerne i databarene.

- **Licens og installation**

Spillet kræver en enkel license, fordi det er så lille et system.
Installationen vil foregå ved at en administrator ved DTU lægger filerne på alle brugere, hvilket gør spillet ikke skal installeres og derfor ikke roder i Windows-filer. Derudover gør det også at spillet ikke skal lægges på en computer ad gangen, hvilket gør processen meget hurtigere.

- **Supplerende funktioner**

1. Spillet skal være på engelsk, og være let at oversætte.
2. Det skal være nemt at skifte terning i koden.
3. Programmet skal kunne køres på DTU's databarer.
4. Spiller og Konto klassen skal nemt kunne bruges i andre programmer.
5. Forsinkelser skal højst være på 333 ms.

- **Begrænsninger**

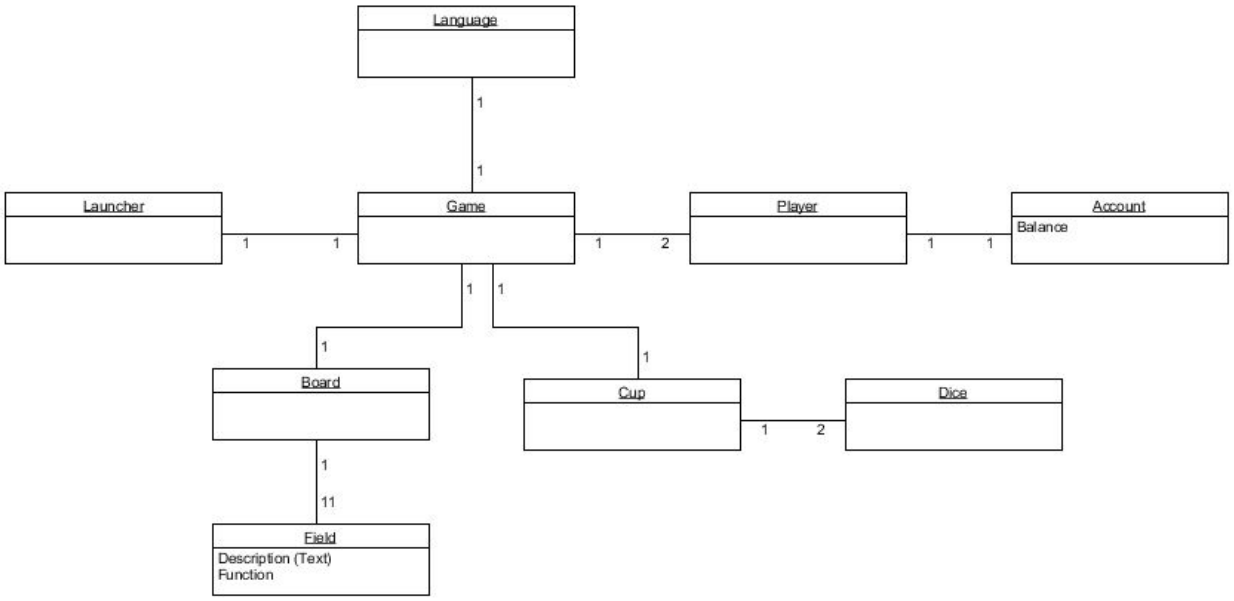
Spillet kan der ikke vælges mere end 1 sprog af gangen, således at en dansktalende og engelsktalende ikke kan spille sammen.
Spillet er lavet specifikt til 2 personer, hvilket betyder at en gruppe på et ulige antal mennesker ikke kan splittes op i grupper af 2, uden at udelukke en fra muligheden for at spille.

- **Juridiske regler**

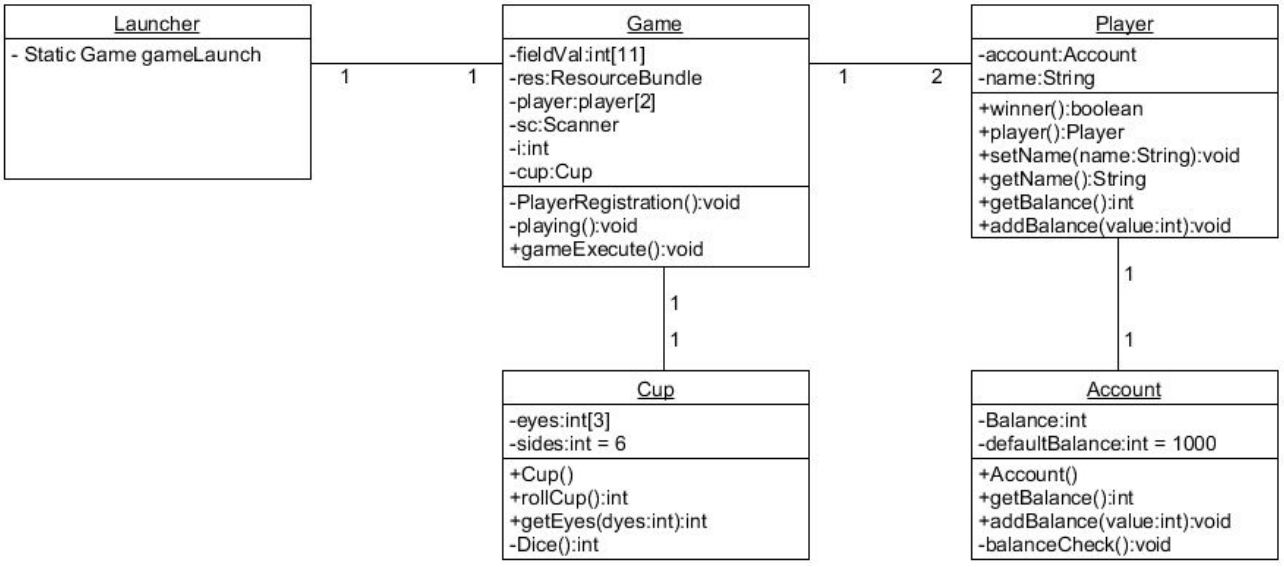
I forhold til regler inden for gambling, samt persondataloven vil spillet ikke få nogle problemer, da pengene i spillet er fiktive og ikke kan bindes til virkelige penge på nogen måde. Brugeren indtaster selv et brugernavn, der ikke bliver gemt til videre brug, og fjernes når spillet slutter.

3 Design

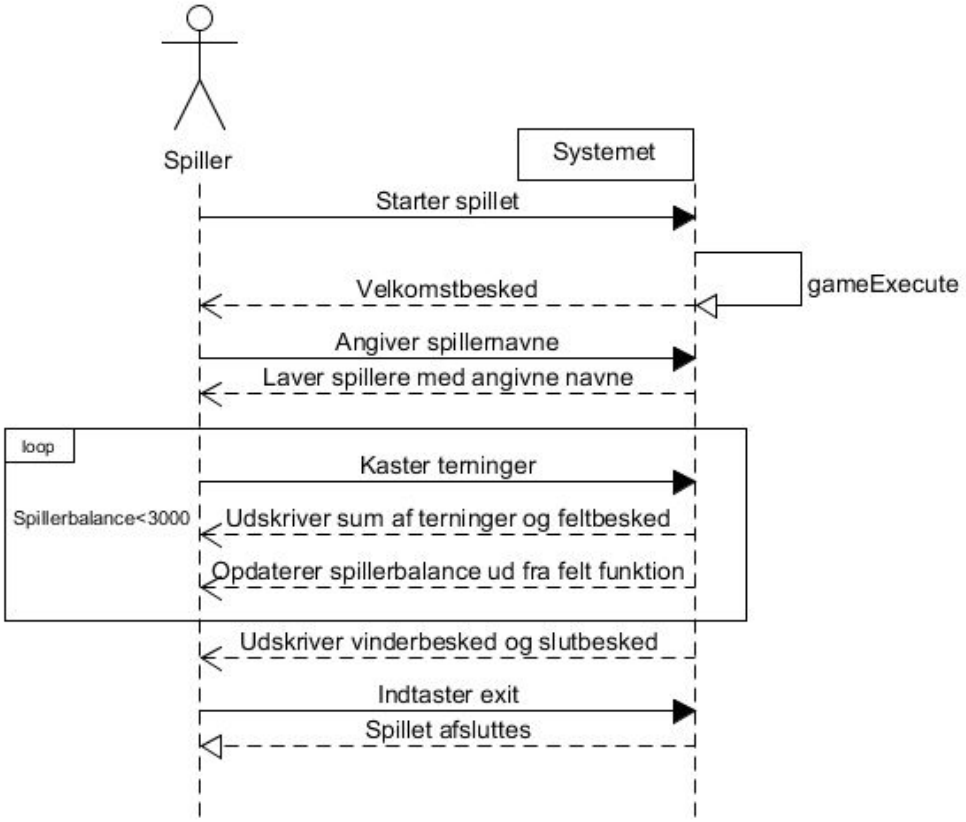
Domænemodel

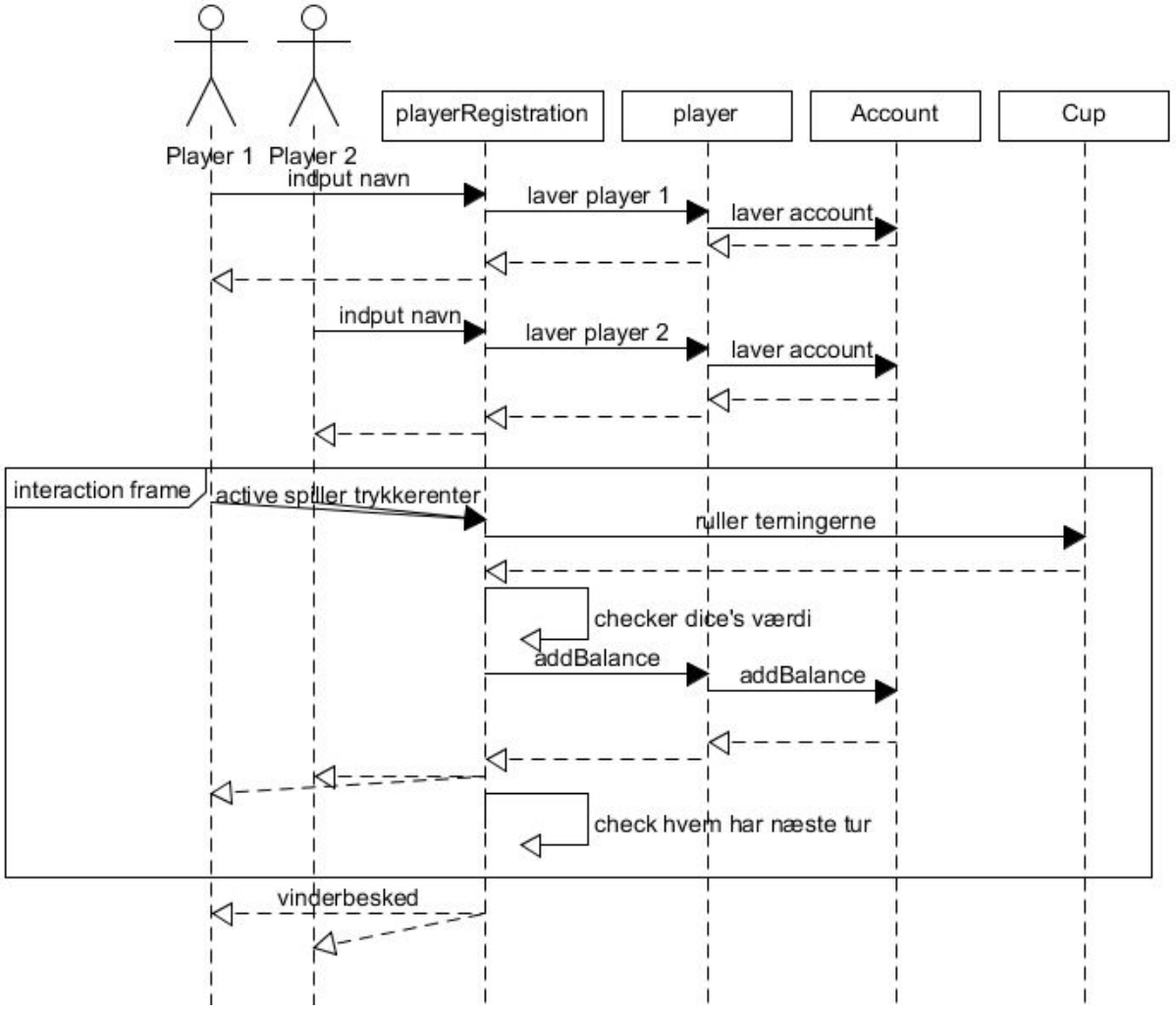


Designklassediagram (DCD)



Systemsekvensdiagram (SSD)





Forslag til design:

- main kører fra Launcher-klassen
    - Bruger vælger sprog
        - Language klassen sender en String[] tilbage
- main kører Game med String[] som input
    - main spørger efter navn på Spiller 1
        - Game laver en Player klasse med navn og en Account
            - Account sætter spillerens balance til 1000
    - main spørger efter navn på Spiller 2
        - Game laver en Player klasse med navn og en Account
            - Account sætter spillerens balance til 1000
    - Game kører -Playing

        - Playing “looper” ture indtil en vinder er fundet
            - Aktiv spiller bruger cup(); til at få en værdi mellem 2-12
            - field(); checker hvad der skal ske, balancen ændres derefter
    - Vinderbesked bliver vist.

GRASP-mønstre beskrivelse:

Launcher klassen kan være creator, fordi den har et ansvar for at oprette et objekt af klassen Game.

Player klassen kan være en information Expert, fordi den indeholder vigtige operationer. Den har ansvar for getName og getBalance og addBalance.

Game klassen kan være controlleren, fordi den har ansvar for at modtage og koordinere flere af systemoperationerne.

Designklassediagrammet viser hvordan klasserne er blevet delt.

4 Implementering

Vi har oprettet 5 klasser: Game, Player, Cup, Account og Launcher, hvor Launcher indeholder main metoden der starter spillet.

Hver af de 5 klasser indeholder forskellige metoder der bruges til spillet, i de forskellige klasser er der:

Account holder styr på spillerens balance, hvor der er metoder til at hente og sætte balancen, samt gøre således at balancen ikke kan gå i negativ.

```
private void balanceCheck() {
    if (this.Balance < 0)
        this.Balance = 0;
}
```

Player holder styr på spillerne og hvornår de kan vinde spillet. Derudover virker klassen også som binding imellem Game og Account, så der skabes lav kobling.

```
public boolean winner() {
    if (account.getBalance() >= 3000)
        return true;
    else
        return false;
}
```

Cup har ansvaret for at bægeret og de terninger, der er i. Derudover gør Cup også at terningerne bliver genstartet, så spilleren ikke lander på felt 9 ved at slå 5 også 4.

```
public Cup(){
    for (int i = 0; i < eyes.length; i++)
        this.eyes[i] = 0;
}

private int Dice() {
    return (int)(Math.random()*sides+1);
}

public int rollCup() {
    eyes[0] = 0;
    for(int i = 1; i < eyes.length; i++) {
        eyes[i] = Dice();
        eyes[0] += eyes[i];
    }
    return eyes[0];

}
```

Launcher gør ikke andet end at gøre spillet statisk samt at køre spillet fra public static void main.

```
public class Launcher {
    static Game game;

    public static void main(String[] args) {
        game = new Game();
        game.gameExecute();
    }
}
```

Game samler alle de andre klasser og er den klasse hvor alt informationen ender. Fra klasserne Player og Account laver Game en spiller med en balance, klassen Cup giver Game informationer om bægeret, således at spillerklassen kan bruge bægeret til at flytte rundt på spillebrættet.

```
do {
    ...
}
while (player[i].winner() == false || player[i].winner() == true);
System.out.println(player[i].getName() + " " + res.getString("winMSG1") + " " + player[i].getBalance() + " "
        + res.getString("winMSG2"));
```

5 Test

Vi har testet vores program på en af DTU's databarer, efter installation af java, som ikke lå på computeren, kunne programmet køres uden problemer. Dette var testet på en "K-Bar", i bygning 341.

Testcases

Test case ID	TS01
Summary	I stedet for at en spillers balance går i negativ, bliver den i stedet 0.
Requirements	K1
Preconditions	Opret new balance "test" med Balance = 1000
Postconditions	Balance = 0
Test procedure	1. Tilføj -1100 til balancen. 2. Check om balancen er 0 eller -100
Test data	test.addBalance(-1100)
Expected result	Balancen er 0
Actual result	Balancen er 0
Status	Passed
Tested by	Cornelius og Casper
Date	10-11-2017, 10:47
Test environment	Eclipse Oxygen 4.7.0 i Windows 10

Test case ID	TS02
Summary	Test af om terningerne er “fair”
Requirements	K3
Preconditions	Opret new Cup med 2 terninger
Postconditions	
Test procedure	1. Udregn mængden af slag baseret på størrelsen af terningen 2. Udregn sandsynlighed for hver terningekast baseret på antal sider på terningerne. F.eks for slaget 12 på 2 6-sidede terninger er sandsynligheden 1/36. 3. Kast Cup 10000 gange. 4. Systemet udregner afvigelse og sammenligner med en procentsats.
Test data	dice_Roller() result_init() expected_calc()
Expected result	Afvigelsen er indenfor procentsatsen
Actual result	Afvigelsen er indenfor procentsatsen
Status	Passed
Tested by	Cornelius og Casper
Date	10-11-2017, 11:44
Test environment	Eclipse Oxygen 4.7.0 i Windows 10

Test case ID	TS3
Summary	Test af om spillet kan køre på computerne i DTU's databarer
Requirements	K11
Preconditions	Java er installeret
Postconditions	Spillet afsluttes
Test procedure	1. Spillet konverteres til en runnable JAR og placeres på databar computerens skrivebord 2. Spillet køres fra databar computeren med kør.bat fra skrivebordet 3. Spillet spilles
Test data	Kør.bat { java -jar spil.jar pause }
Expected result	Spillet kan køre på computeren
Actual result	Spillet kan køre på computeren
Status	Passed
Tested by	Kasper
Date	10-11-2017, 13:11
Test environment	Eclipse Oxygen 4.7.0 i Windows 10

6 Brugervejledning

Du downloader .zip filen, og derefter udpakker den med enten dit styresystems udpakningssoftware, eller med 7-zip, hvilket du gør på deres hjemmeside: http://www.7-zip.org/download.html . For at åbne programmet, skal du have installeret den nyeste version af Java, hvilket du gør på Oracles hjemmeside: https://java.com/en/download/ . Når du har downloadet og installeret Java på din computer, er du klar til at køre spillet. Spillet køres, du indtaster de to spilleres navne, og kaster terningerne ved at trykke på enter-knappen.

I forhold til minimumskrav skal spillet kunne køre på computerne i databarerne, samt at der skal være installeret Java Runtime Environment, hvilket er det der får JAR-filen til at fungere. For at kildekoden kan blive til en JAR-fil og køres skal den først compiles.

En compiler er et program, der fungerer som en oversætter, der kan oversætte kildekoden, der kan læses af mennesker til kode der består af bytekode, hvilket en computer kan læse. Imens dette sker ændres filnavnet fra .java til .class. JAR er en masse klassefiler, der interagerer med hinanden.

I forhold til dette spil er der ikke brug for en installering, da JAR kører direkte i Java, hvorimod andre filer og programmer først skal installeres fordi de skal kommunikere med systemet direkte. F.eks. hvis der skal åbnes en hjemmeside skal styresystemet åbne browseren osv. Endeligt kan kildekoden afvikles, i dette tilfælde via JAR-filen.

Kildekoden kan også gå i den modsatte retning, f.eks. hvis det skal hentes fra et git repository, hvor alle de importerede filer kan være lagt som JAR eller .class fil(er). Her vil compileren oversætte den modsatte vej og oversætte fra noget computeren kan forstå til noget mennesker kan forstå.

7 Projektplanlægning

Opgavebeskrivelsen beskrev en tilgangsmåde, som var den anbefalede rækkefølge at tilgå projektet. Denne tilgangsmåde har vi forsøgt at bruge til det fuldeste, med få undtagelser, hvor systemsekvensdiagrammet var en smule bagud, samt at rækkefølgen i programmeringsdelen var lidt anderledes.
Vi har ikke anvendt nogen som helst tidsplanlægningsværktøjer, da vi ikke så at dette var nødvendigt med et mindre projekt som dette.
Derudover har vi anvendt et arbejdsdokument (Se bilag), som beskriver alle punkterne i projektet, hvem der skal lave de forskellige punkter, om man er i gang med punktet, og om punktet er færdigt. Dette brugte vi til at holde styr på strukturen i projektet, så vi ikke begår samme fejl som i sidste projekt, hvor to laver det samme, uvidende om at den anden laver det samme.

Desuden har vi også anvendt github, til versionsstyring af vores kode, samt brugt det til at få en bedre struktureret projekt.

8 Konklusion

Vi kan konkludere at processen har været forbedret i forhold til sidste projekt. Dette skyldes højst sandsynligt, at vi har lavet mere projektplanlægning, iht. et arbejdsdokument, og en mere struktureret fremgangsmåde til projektet. Dette kan man også se på vores endelige produkt, da vi har nået at lave det hele, dog har vi valgt ikke at anvende GUI'en i dette projekt.
Samtidigt har vi valgt at anvende en ResourceBundle, som vi valgt at bruge til nemt at kunne skifte sprog, samtidigt vælger den også automatisk sprog udefter sproget på computeren.
Vi kan desuden også konkludere, at i forhold til tidligere projekter, har dette projekt forløbet meget bedre. Arbejdsopgaverne har været bedre uddelt, så enhver i gruppen har haft chancen for at være inde under alle områder, hvilket var et mindre problem i vores tidligere projekt.
Fordelingen i programmeringsdelen kunne dog godt have været bedre, hvilket vi vil optimere til næste projekt.
Vores brug af Github har også været bedre struktureret end sidste gang, så vi har sparet tid på diverse fejl og rettelser i de forskellige repositories.

Litteraturliste

Bilag 1 (Gruppekontrakt)

Vi mødes hver fredag efter versionsstyring (Ikke mellem 12 og 13) eller 13:00 for at organisere projektet og give individuelle opgaver for hvad vi skal nå for denne uge.

Alle skal være forberedt til det pågældende møde.

Man må ikke møde op fuld, eller med tømmermænd!

Hvis man ikke kan komme til mødet, skal man sætte sig ind i tingene hjemmefra og læse op på referatet af mødet, så man er indforstået med beslutningerne.

Man skal udmelde udeblivelse til resten af gruppen, så snart det er muligt, helst dagen før via facebook gruppen.

Navn	Telefonnummer	Email
Thomas	29728727	Thomaslvestergaard@gmail.com
Casper	51334473	Casperkandersen@gmail.com
Kasper	25778122	huez@rocketmail.com
James	60670689	jamesgileskbh@gmail.com
Cornelius	30130595	hedegaard.cornelius@gmail.com
Tahany Abdi	27851038 42746095	tahanynm@gmail.com Abdirizaqfarah@gmail.com

Bilag 2 (Arbejdsdokument)

ARBEJDSDOKUMENT

// MÅ GERNE ÆNDRES I \\

SKIFT FRA RØD TIL GRØN HVIS FÆRDIGGJORT!!! (Copy grønt felt=>Paste i rødt felt)

HUSK AT SKRIVE JERES NAVN I "I gang" NÅR I ER I GANG OG SLETTE NÅR I IKKE ER!!!

Rapport (Analyse/design):	Hvem:	I gang:	Kommentarer:	
Kravliste/Kravspecifikation	Alle		Kravliste lavet, må gerne opdateres ligger i rapport	
Inddele kravene i FURPS+	Kasper			
Use case diagram	Alle		Use case diagram lavet, men kan altid ændres.	
Use case beskrivelser brief	Casper			
Fully-dressed beskrivelse af 1 central use case	Casper			
Domænemodel	Alle		Domænemodel lavet, og diskuteret med underviser.	
Systemsekvensdiagram	Kasper		Har lavet, men kan godt eftertjekkes	
Designklassediagram	Cornelius			
GRASP-mønstre beskrivelse	Tahany			
Vision	Casper			
Indledning	Kasper, Thomas			
Sekvens diagram	Cornelius			
Konfiguration	Casper			
Implementering	Tahany			
Brugervejledning	Kasper			
Aktør/Interessentanalyse	Thomas			

Projektplanlægning	Kasper			
Konklusion	Kasper			
Tests-cases	Cornelius			
Tilrettelse og stavekontrol	Thomas			

Programmering:	Hvem:	I gang:	Kommentarer:	
Lav en klasse Player	Tahany, Kasper, Thomas			
Lav en klasse Account, som beskriver spillerens penge	Cornelius			
Lav .properties fil, både default og engelsk.	Thomas		Brug: https://docs.oracle.com/javase/6/docs/api/java/util/ResourceBundle.html	
Lav klasse: Game	Cornelius			
Lav klasse: Launcher	Kasper, Thomas		Lavet main, men skal have nogen rettelser og nogle tilføjelser.	
Lav klasse: Cup	Casper			
Skrive kommentarer	Casper			

Test:	Hvem:	I gang:	Kommentarer:	
JUnit test, som beviser balance aldrig kan blive negativ	Tahany			
JUnit test2, cup: dice metode er random, 2d12 er random, 2dX er random.	Cornelius			