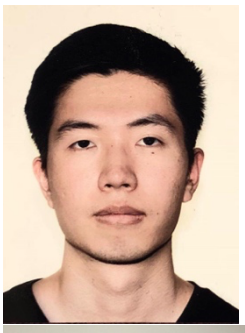
02312 62531 62532

INDLEDENDE PROGRAMMERING, UDVIKLINGSMETODER TIL IT-SYSTEMER OG VERSSIONSTYRING OG TESTMETODER

CDIO 1

Gruppe 22

Andreas Vilholm Vilstrup
s205450



Chenxi Cai
s205420

Alexander Solomon
s201172

Oliver Fiedler
s205423

Isabel Grimmig Jacobsen
s205473

Ahmad Shereef
s173750

Afleveret d. 30/10-2020

Resumé

Vi har fået til opgave at lave et spil for spilfirmaet spilfirmaet IOOuterActive. Det har opstillet en række krav og visioner som vi i samarbejde har analyseret specificeret og udviklet. De givne krav er at finde i afsnittet af samme navn.
Denne rapport dokumenterer udviklingen af det tidligere omtalte spil. Jf. afsnittet om krav spilles dette af to personer, som skiftevis slår med to terninger og lander på felter, som har forskellige værdier. Begge spiller har hver en pengebeholdning på 1000 kroner ved spillets start. Pengbeholdningen kan blive større og/eller mindre i takt med at spillerne lander på de forskellige felter. Der er i alt 11 forskellige felter, som enten tilføre eller fratrækker penge til spillernes individuelle pengebeholdninger. Spillerne vinder spillet ved at, være det første der når en pengebeholdning på 3000 kroner.
Denne rapport indeholder metoder og diagrammer, der er med til at oplyse hvordan selve spillet er opbygget. Der er bl.a. Lavet et use case diagram, som viser hvilke muligheder spillerne har og hvad de skal gøre. Der er også lavet et Flowchart diagram, som viser og fortæller om programmets processer virker.
Til at teste selve programmeringen/ koden er der benyttet JUnit, til at være sikker på at spillet kører uden problemer. Vi har bl.a sikret at pengebeholdningen ikke kan komme under 0 og at hver terningerne ikke kan slå mere end 6.
Ud Fra diagrammer, metoder og test af programmeringen/koden, kan vi konkludere at spillet er let at spille og forstå. Spillet er lavet sådan at man som spiller hele tiden bliver fortalt hvad man skal gøre, derudover får man også en introduktion til reglerne og hvad det indeholder, i starten.

Indholdsfortegnelse

Indledning

I denne rapport dokumenterer vi en opgave, som er blevet stillet af spilfirmaet IOOuterActive. Denne opgave er en videreudvikling på den forrige opgave CDIO del 1, og har derfor vil denne rapport indeholde noget af det samme.
Opgaven går ud på at videreudvikle et terningespil, så det bliver til en spil med forskellige felter. Kunden har stillet nogle krav, som vi har prøvet så vidt muligt at implementere.
Spillet foregår mellem to spillere, som skiftes til hver at slå med to terninger og rykke mellem 11 forskellige felter. Antallet af terningernes øjne afgør hvilket felt man lander på, som indeholder forskellig positive eller negative penge belønninger. Den spiller der først opnår et pengebeløb på 3000 kroner, vinder spillet. Derudover vil kunden gerne have at man starter med en pengebeholdning på 1000 kroner.
Rapporten indeholder også modeller og diagrammer, som er med til at give et overblik og en forståelse for hvordan spillet virker og hvad det indeholder.

Krav

Kravlisten er delt op i to dele, funktionelle krav og ikke-funktionelle krav. Funktionelle krav er ting programmet skal kunne. Ikke-funktionelle krav, kan være kvalitetskrav, krav til videre udvikling eller skalerbare krav.

Funktionelle krav.

Hver spiller skal kunne slå med to terninger.
Terningslag skal give en værdi som giver spilleren en position.
Spillet er slut når en spiller når 3000 point/kroner.
Point vises med det samme efter hvert slag.
Spiller skal kunne tjene og miste point/penge.
Hvert givet felt skal give en vis mængde point.
Spillerne starter med en pengebeholdning på 1000.
Pengebeholdning må ikke blive mindre end 0.
Tekst om feltet skal udskrives når man lander på det.

Ikke-funktionelle krav.

2 spillere.
2 terninger.
Skal kunne spilles på DTU's maskiner.
Spiller og pengebeholdning skal være en klasse for sig selv.

Vurdering af kundens krav og visioner

Der er en række af kundens krav som enten ikke er specificeret eller ikke uddybet i en sådan grad at det ikke er til at dokumentere hvorvidt det mødes eller ej.

Spillet skal let kunne oversættes til andre sprog.

Dette krav er ikke specifikt nok. Dette giver en stor grad af frihed i forhold til hvordan dette bliver mødt. Det vil være en nødvendighed at specificere om dette vil være i form af en mulighed for at vælge sprog ved starten af spillet, eller om sproget blot skal være simpelt nok til at man hurtigt kan oversætte det til et andet sprog. Altså vil det være en udfordring at konkludere hvorvidt kravet er blevet mødt eller ej.

Det skal være let at skifte til andre terninger.

Dette krav er heller ikke specifikt nok. Kravet giver heller ikke mening, da terninger vil altid være det samme, uanset hvilken slags terning man bruger, så længe det er en 1-6 terning. Da det ikke er et målbart krav er det svært at implementere. Derudover vil andre terninger end en 6-sidet ikke fungere indenfor spillets rammer.

Spillerne lander på et felt numrene fra 2-12.

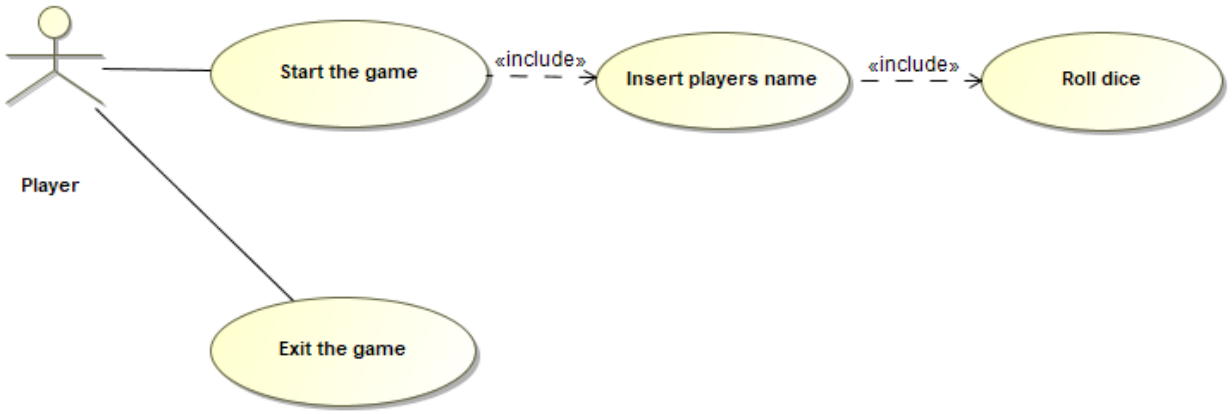
Det er ikke specificeret hvorvidt spillerne skal blive på det felt de lander på, eller om det starter på et imaginært felt, som giver det samme startpunkt hver gang.

Det kan dog med nogen sikkerhed siges at dette ikke er tilfældet, da der så ville være behov for at have et felt nummer 1.

Analyse

Use case diagram.

Use case diagrammet nedenfor viser hvad den enkelte spiller skal/kan gøre, derudover viser det også hvad der sker/ er inkluderet når spilleren udføre noget.



Figur 1: Use case diagram fra gruppe 22 CDIO del1

Use case brief beskrivelse:

Use case	Beskrivelse
Start spil	Tryk enter for at starte spillet og komme til næste funktion.
Indtast navn	Programmet vil have du skriver spillerenes navne for at komme videre
Kast terningen	Tryk enter for at få terningerne til at rulle og derefter vise et antal øjne, som er det antal felter du skal flytte.

Use case Fully dressed beskrivelse:

Use case section	Comment
Use case name	Indtast navn
Scope	Nu vil spillet bede dig/jer om at indtaste hver jeres navn, dvs Spiller 1 og Spiller 2 indtaster begge et valgfrit navn. Først vil spiller 1 blive bedt om at indtaste navn, som godkendes ved at taste enter. Det samme gentages nu for Spiller 2. Når spiller 2 godkender navnet, vil man blive ledt videre til en ny del af spillet.
Level	sub-function
Primary actor	Spiller

Success guarantee	char/ string bliver intastet for hver spiller, så det ikke er tradiontenelle navne
Main success scenario	Indtastning af spillernes navne
Extensions	Ved tekniske problemer programme starte forfra.
Special requirements	De skal kunne skrive 2 navne
Technology and data variations list	-
Frequency of Occurrence	Test af programmet kan fortages på en ugentlig basis
Miscellaneous	Hvad sker der hvis navnet ikke indtastes efter et stk. tid?

Use case section	Comment
Use case name	Start spil
Scope	Tryk enter for at komme i gang med spillet, og derefter vil man blive sendt videre til en ny side.
Level	Bruger mål
Primary actor	Spiller
Success guarantee	Når spilleren har trykket enter vil det medfører et succes scenario, da programmet vil kunne fortsætte til næste del af spillet.
Main success scenario	At trykke enter og komme videre i spillet
Extensions	Ved tekniske problemer programme starte forfra.
Special requirements	Det er kun enter knappen der har en funktion
Technology and data variations list	-
Frequency of Occurrence	Test af programmet kan fortages på en ugentlig basis
Miscellaneous	Hvad sker der hvis man trykker på andre taster end enter?

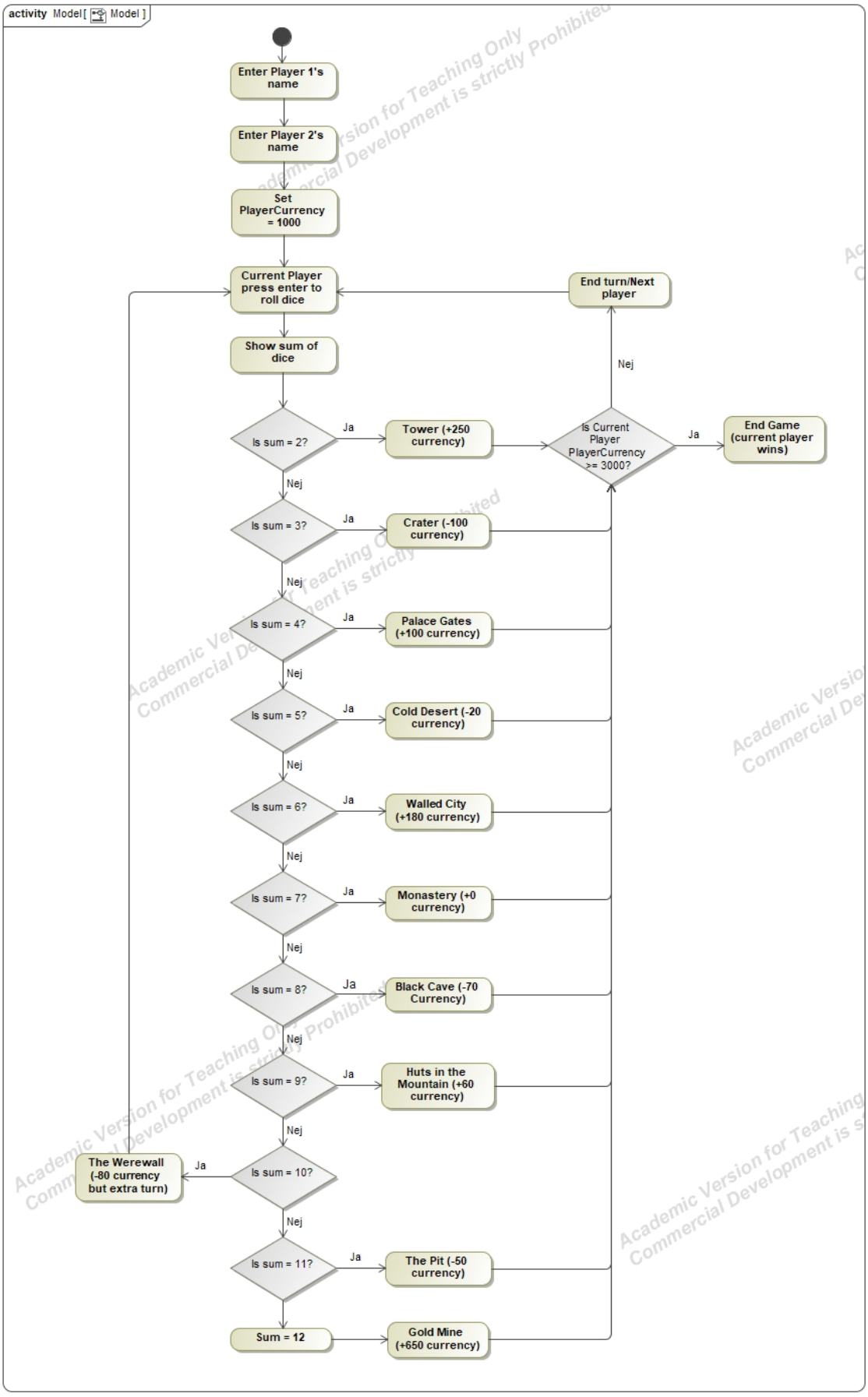
Use case section	Comment

Use case name	Kast terningerne
Scope	Spillet vil nu bede Spiller 1 om at trykke enter, for at kaste terningerne og derefter vil antallet af øjne vise hvor mange felter du skal flytte frem. Når turen er slut bliver Spiller 2 bedt om det samme og på den måde fortsætter spillet indtil den ene spiller opnår 3000 point.
Level	sub-function
Primary actor	Spiller
Success guarantee	Begge terninger viser deres øjne, hvilket vil resultere i en sum mellem 2 og 12. Og der vil derefter blive tildelt point baseret på det felt som terningerne henviser til.
Main success scenario	Summen på de to terninger bliver vist
Extensions	Ved tekniske problemer programme starte forfra.
Special requirements	Det er kun enter knappen der har en funktion
Technology and data variations list	-
Frequency of Occurrence	Test af programmet kan fortages på en ugentlig basis
Miscellaneous	Hvad sker der hvis den kun kaster en terning? Hvad hvis den ikke viser summen af terningerne?

Flowchart

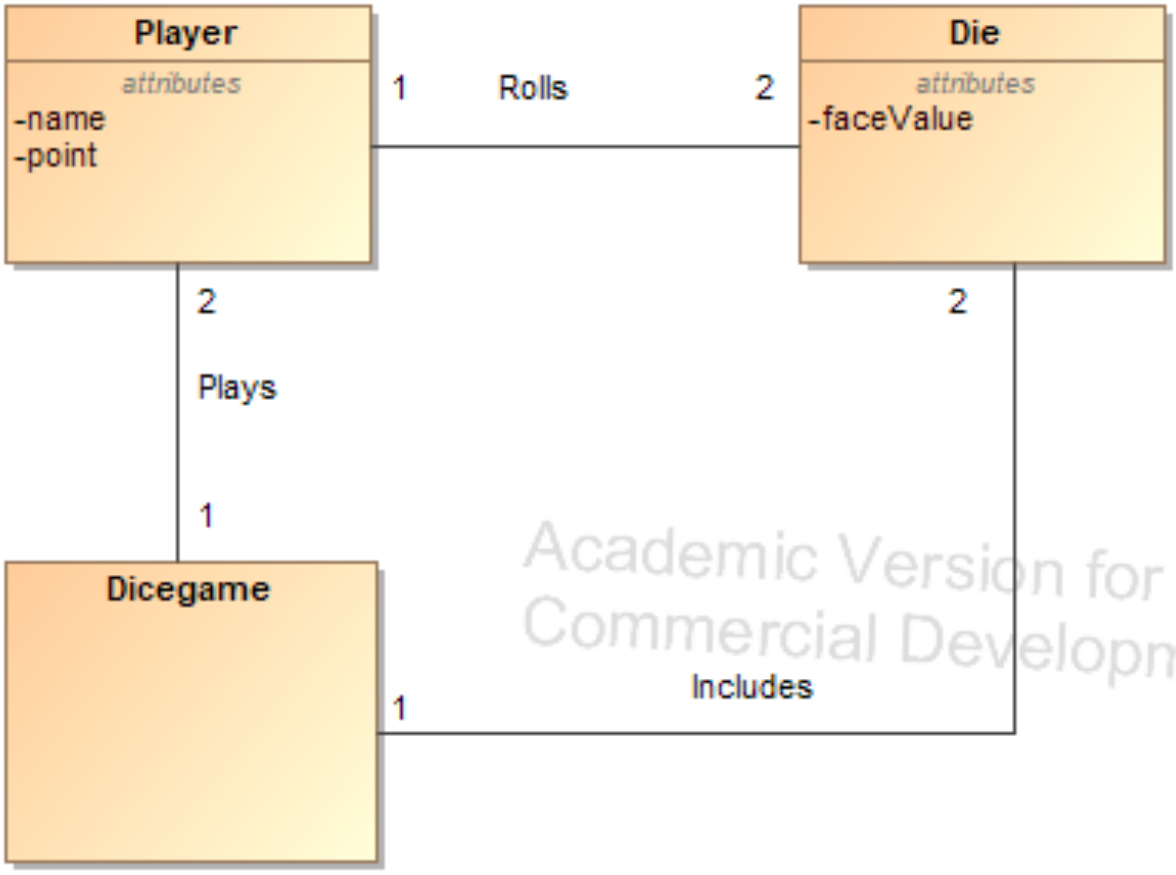
Nedenfor ses flowchart over programmet. Her kan man aflæse hvordan systemet kommer til at virke, og dens processer.

Det kan ses, systemet starter med indtastning af spillernavne, hvorefter systemet sætter hver spillets konto til dens startværdi 1000. Derefter kan spillet begynde hvor det beder den første spiller om at trykke på “enter”, for at slå terningerne. Hver gang terningerne bliver kastet, vil systemet tjekke tjekke summen og ud fra det, fortælle spilleren hvilket felt den er landet på og hvor mange penge det felt vil lægge til/trække fra spillerens konto.
Efter den process vil spillet gå videre til den næste spillers tur, medmindre man får terningsummen 10, som vil give en ekstra tur til spilleren.
Systemet vil altid tjekke “currentPlayer’s” konto inden den starter en ny tur. Hvis spillerens konto er kommet over 3000, ville det resultere i at spillet slutter og der vil blive kåret en vinder.


activity Model [Model]
Enter Player 1's name
Enter Player 2's name
Set PlayerCurrency = 1000
Current Player press enter to roll dice
Show sum of dice
Is sum = 2?
Ja
Tower (+250 currency)
Is Current Player PlayerCurrency >= 3000?
Ja
End Game (current player wins)
Nej
End turn/Next player
Nej
Is sum = 3?
Ja
Crater (-100 currency)
Nej
Is sum = 4?
Ja
Palace Gates (+100 currency)
Nej
Is sum = 5?
Ja
Cold Desert (-20 currency)
Nej
Is sum = 6?
Ja
Walled City (+180 currency)
Nej
Is sum = 7?
Ja
Monastery (+0 currency)
Nej
Is sum = 8?
Ja
Black Cave (-70 Currency)
Nej
Is sum = 9?
Ja
Huts in the Mountain (+60 currency)
Nej
Is sum = 10?
Ja
The Werewall (-80 currency but extra turn)
Nej
Is sum = 11?
Ja
The Pit (-50 currency)
Sum = 12
Gold Mine (+650 currency)

Domæne model

Nedenfor ses der en domain model, som viser de klasser vi har, hvor deres attributter også vises. Vi har 3 klasser, *"Player", "die", "Dicegame"*, hvori modellen viser deres relationer til hinanden. Der ses mulitiplicitet indikator mellem klasserne, for eksempel en *Dicegame* har to *Player,* og en *Player* har 2 *Die*.



Design

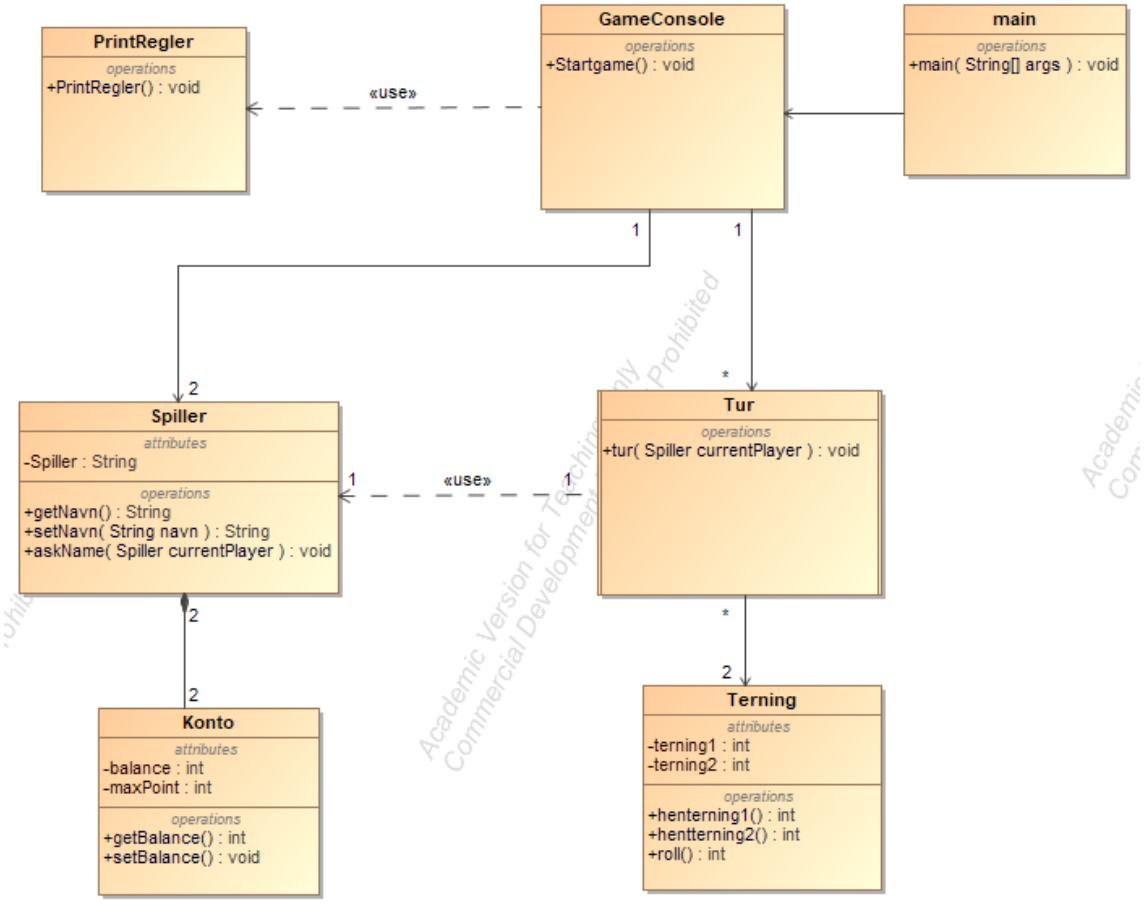
Design KlasseDiagram.

Design klassediagrammet der er blevet dannet nedenfor, viser hvilket relationer og attributter vores klasser har.
Vores main klasse kender til gameConsole uden at gameConsole kender til vores main.
Klassen gameConsol bruger klassen printRegler, men er ikke afhængig af den.
GameConsole indeholder et spil der med et vinderscenarie.

Gameconsole er assoiceret til spiller og tur, hvilket vil sige at gameconsole kender til spiller og tur klassen, uden at de klasser kender til gameconsole.

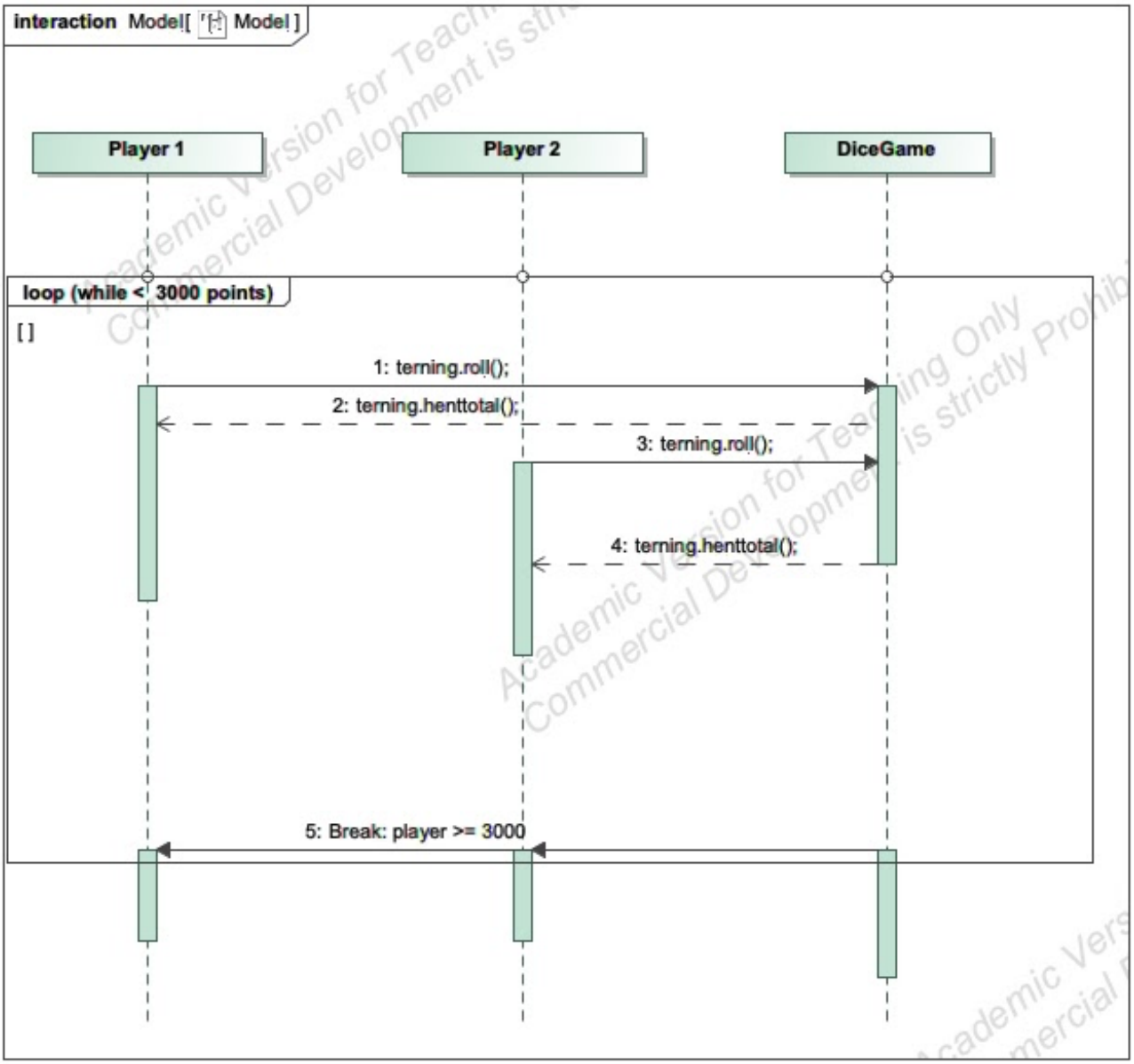
Hver tur benytter sig af spiller klassen, og selve tur klassen kan have ubegrænset ture i spillet, men hver spiller har en tur ad gangen.
Tur klassen benytter sig også af en uafhængig terningklasse, som har 2 terninger der kan slå mellem 2 og 12.

Klassen spiller indeholder spiller navne og ask name funktionen, der kan være 2 spiller i et spil.
De 2 spiller her hver en konto tilknyttet.
Selve spillet indeholder 2 kontoer, med max point på 3000kr, og minimum på 0kr.
Hver spillers konto bliver opdateret efter hver tur, balancen enden vil blive tilføjet eller fratrukket et beløb.



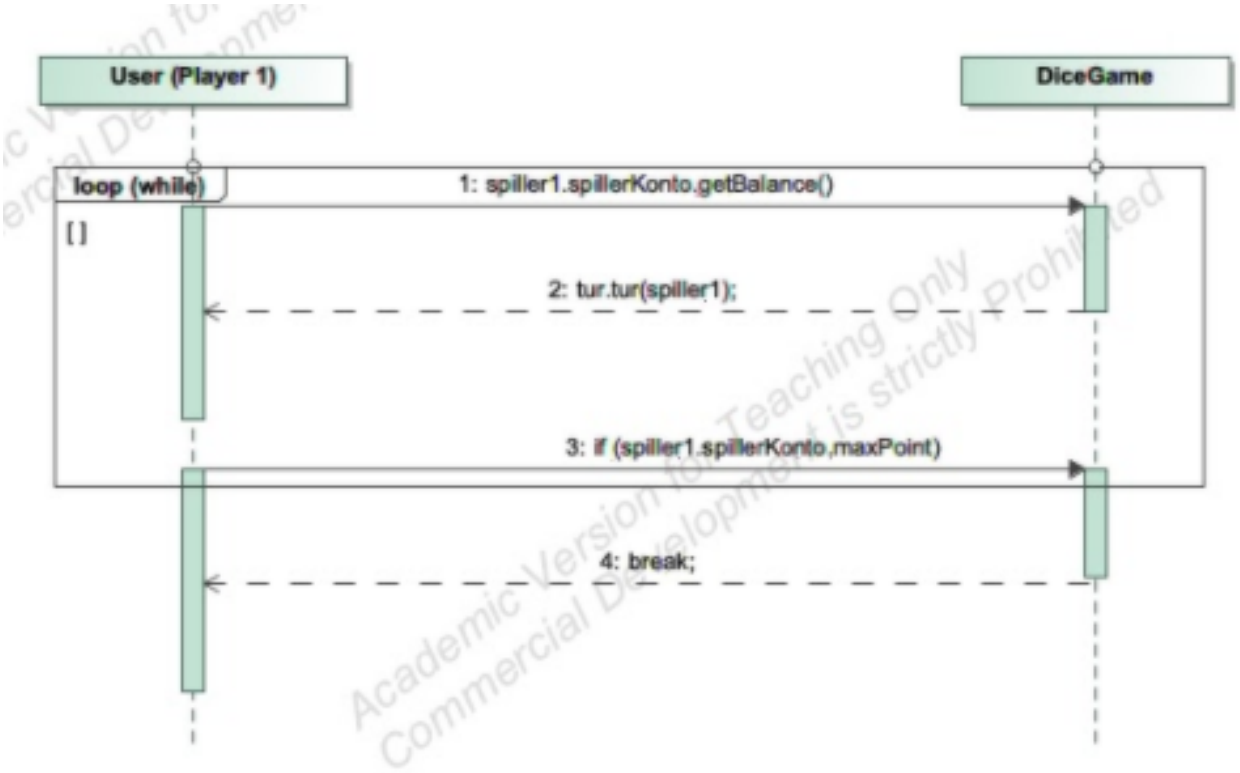
Sekvens diagram

I diagrammet nedenfor ses spillernes interaktioner med systemet, spiller 1 starter spillet ved at kaste med terningen, hvorefter systemet returnerer den samlet værdi til spilleren, og med samme princip, når spiller 2 kaster med terningen. Denne ”while” løkke vil køre så længe spillerens samlet point, er under 3000, i tilfælde af spillerens konto overstiger de 3000 point, vil spilleren bryde løkken og spillet dermed slut.



System Sequence diagram

I dette diagram bliver der beskrevet et bestemt scenarie af en use case i spillet, ”while”-løkken vil køre med brug af metoden ”spiller1.spillerKonto.getBalance()” til at få point, og derefter vil ”get” metoden returnerer point til spilleren. Igen vil spilleren bryde løkken hvis pointene overstiger det maksimale med brug af metoden ”if (spiller1.spillerKonto.maxPoint).



Implementering

En guide til når man modtager filen og vil køre den:

Når man modtager zipfilen downloader man den til et sted hvor man kan finde den igen. Så går man ind på IntelliJ's startside (hvor man kan create new project eller import project osv). Så trykker du på "Import Project" og finder din downloadede zipfil og åbner den. Derefter popper der nogle beskeder. På dem skal du bare trykke næste og acceptere alt hvad den beder om indtil den til sidst lader dig trykke "Finish".

Nu burde du have projektet åbnet på IntelliJ. For at åbne klasserne skal du trykke på pilen ved siden af mappen med navnet 22_del2, så klikke på pilen ved siden af mappen src. Herfra kan du se pdf filen hvor vores rapport ligger. Du burde herfra også kunne se alle vores klasser ligge ved siden af. Dem dobbeltklikker du så for at åbne dem.

For at køre programmet skal du så åbne den klasse der hedder Main og trykke på den grønne pil der står ud for public class Main på linje 1.

Her kan du støde ind i et problem hvor den siger at du bruger en for gammel version af SDK/JDK. Hvis det er problemet skal du trykke på File->Project Structure og så tjekker du om du har en version som er 12 eller højere, oppe i højre hjørne lige under et + og et -. Hvis ikke skal du gå på google og søge "java jdk" og gå ind på det første link og downloade en nyere JDK og følge guiden på hjemmesiden. Så går du ind på Project structure igen og trykker på + og finder din downloadede JDK og apply den så fjerner du din gamle version så kun den nye står tilbage. Så trykker du apply.

Næste gang du så kører programmet kan du risikere den stadig siger fejl og der kommer en meddelelse hvor den beder dig opdatere. Tryk på den, lad den opdatere og så burde det hele virke.

Beskrivelse af vores klasser

Vi har oprettet klasser til at kategorisere vores kode. Klasserne er også oprettet for at holde vores main til et minimums antal kodet linjer. Vi har i alt oprettet syv klasser: Main, Tur, Terning, Spiller, Printregler, GameConsole og Konto.

Tur-klassen: Her skriver vi alt ind hvad én tur bør indeholde og gør det nemt for os at holde struktur på hver spillers individuelle tur. Denne klasse samarbejder med klassen Spiller, main, og konto.

Terning-klassen: Terning klassen er vigtig fordi den viser en tilfældig sum, som viser hvilket felt spilleren lander på og afgør hvor mange point spilleren derefter får. Det gode ved at have denne som klasse, er at vi nemt kan tilføje flere terninger og nemt kan ændre på antallet af øjne hver individuel terning kan have.

Spiller-klassen: Spiller klassen er god at have da vi med den nemt kan indskrive hvor mange spillere der skal være i vores spil og bede om deres individuelle navne som den så vil huske resten af spillet. Den er også vigtig, da den hjælper med at holde styr på hvem der har hvor mange penge.

Spiller-klassen er designet således at enhver spiller har en konto og at denne bliver oprettet sammen med spilleren i konstruktøren. Dette gør det nemt at adresserer lige præcis den spillers balance som man ønsker uden at risikerer at koden bliver ustruktureret.

Printregler-klassen: Denne klasse sparer bare en masse plads i main-klassen, da man her skriver en masse printlines som fylder en del og dem kan man så bare sætte ind som en enkelt linje kode inde i main.

Konto-klassen: Her kan vi holde styr på hvor mange penge der ligger på hver spillers konto.
GameConsole: Her ligger vi elementerne fra alle de andre klasser ind i rækkefølge som er nødvendigt for at kører spillet. Det er også her vi får printet reglerne, skriver det antal spillere vi gerne vil have, spørger spillerne hvad de hedder, kører igennem deres tur og stopper spillet hvis en spillers konto rammer 3000 kr.

Main-klassen: Herfra kører vi hele projektet ved at indsætte GameConsole

Test

I forbindelse med test af programmet er der dels blevet benyttet JUnit, og dels er der blevet testet ved at fremtvinge specifikke scenarier ved gentagen gennemgang af koden i praksis.

GameConsole er blevet gennemgået, så alle betænkeligheder og scenarier er blevet testet.

JUnit er blevet brugt i forbindelse med test af Terning.java klassen og Konto.java klassen.

Terningerne bliver testet således at det ikke er muligt at slå mere end 6 og mindre end 0. Derudover er der også blevet tilføjet en funktionalitet der viser normalfordelingen af terningeslagene over 1000 kast, hvis dette ønskes.

Konto.java er blevet testet således at det kan bekræftes at kontoens balance ikke kan blive mindre end nul.

Kontoen er opbygget således at hver konto har en integer der hedder balance og en integer der hedder maxPoint. Balancen er bygget op omkring en getter- og setterfunktion. Denne funktionalitet giver muligheden for at få balancens nuværende værdi(getter) samt at ændre i kontoens balance(setter).

Ved test af kontoen, er det et krav at kontoens balance ikke kan blive mindre end 0. Derfor at der ved hjælp af JUnit blevet lavet en test der forsøger at få balancen til at blive mindre end nul. Ved udførelse af denne test kan det konkluderes at dette ikke er muligt.

Projektplanlægning

Intellij:
Intellij bliver i kodningen benyttet til at kode i Java. Derudover tilbyder Intellij en god integration af Github. Dette må anses som værende svært nødvendigt med et multiplum af mennesker arbejdende på dette projekt.

Github:
Github bliver benyttet i projektet til at give flere personer mulighed for at arbejde på det samme projekt. Derudover giver det mulighed for versionsstyring i forbindelse med gendannelse af ældre versioner samt at kunne holde styr på hvem der laver hvad. Derudover giver det også mulighed for branching.
https://github.com/chen0046/22_del2

Use case:
Vi har benyttet use case til at give en oversigt af vores aktører, og hvad systemet skal kunne, dermed er den også med at illusterere forholdet mellem aktører og casene.

Domain model:
Vi har lavet en domain model, da den giver et overordnet blik over vores klasser og hvilken attributter de har. Derudover kan man se relationer mellem klasserne. Den er forholdsvis simpel at se på, derfor har vi valgt at lave den, da den vil give kunden et overblik over vores system.

Flowchart:
Et flowchart er en visuel præsentation af hvordan koden kommer til at fungere. Dette viser step-by-step beslutningsprocessen for programmet og de parametre den arbejder indenfor. Dette laves i høj grad for at simplificere programmeringen i forbindelse med implementeringen.

System sekvensdiagram:
Et systemsekvensdiagram bliver brugt til at vise et bestemt scenarie i en use case, de situationer som eksterne aktører skaber samt deres rækkefølge. Dette diagram bliver ofte brugt til at visualisere den succesfulde use case eller eventuelle komplicerede alternativer.
I dette diagram er der betydelig fokus på hvordan forskellige klasser og metoder interagerer med hinanden og hvordan grænserne krydses. Det er derfor også en effektive metode at vise forskellige klassers metoder benyttes og i hvilke sammenhænge.

Design klassediagram:
Vi har lavet en design klassediagram til kunden, da den giver et overblik over vores system og hvilken/hvor mange klasser vi har, dermed kan man også se deres attributter og metoder.

Sekvensdiagram:

Et sekvensdiagram bruges til at vise samarbejdet mellem objekter. I et sekvensdiagram kan man se når en aktion bliver lavet fra et objekt til et andet og om den aktion kræver at det ene objekt kan arbejde videre eller er nødt til at vente på et respons efter dens aktion. Kunden vil kunne bruge dette til nemt at få et overblik over hvilke objekter vi arbejder med, hvordan de arbejder sammen, rækkefølgen de arbejder i og afhængigheden mellem objekterne.

JUnit:

JUnit er et testprogram man specifikt bruger til programmeringssproget java. JUnit er en en del af gruppen xUnit, som er en samling af andre test framworks. Alle frameworks der ligger i gruppen xUnit, bliver navngivet efter, hvad forbogstavet for programmeringssproget er (et stort bogstav), efterfulgt af Unit (stort U)

Konklusion

I denne opgave har vi videreudviklet det terningspil, som vi også arbejdede med i CDIO del 1. Opgaven er løst med hensyntagen til en række krav, stillet af kunden IOOuterActive, som har sat rammen for spillet.
Terningspillet kan spilles af to personer. Spillerne slår skiftevis med to terninger og rykker hen til det felt, som fremstår af øjnenes samlede værdi. Hvert felt på spilfelterne har forskellige værdier, som både kan være positive og negative - Fx +300 eller -200, i alt 11 forskellige felter. Hver spiller har en pengebeholdning, som starter på 1000 kr., og som bliver større eller mindre afhængigt af felterne, som den enkelte spiller lander på. En spiller kan ikke skylde penge, hvilket vil sige at den individuelle pengebeholdning aldrig kan blive mindre end 0 kr. Den første spiller der når 3000 kr., vinder spillet.
Spillet er testet ved hjælp af JUnit, for at sikre kvaliteten og rette eventuelle fejl, og indeholder desuden en guide, så det er nemt at komme i gang.

Timeregnskab.

Tidsforbrug i antal minutter					
UGE	43				
Navn/Dato	19/10/2020	20/10/2020	21/10/2020	22/10/2020	23/9/2020
Alexander Solomo	0	0	220	250	160
Andreas Vilholm V	0	0	220	250	160
Ahmad Shereef	0	0	220	250	160
Chenxi Cai	0	0	220	250	160
Isabel Grimmig Ja	0	0	220	250	160
Oliver Fielder		0	220	250	160
UGE	44				
Navn/Dato	26/10/2020	27/10/2020	28/10/2020	29/10/2020	30/10/2020
Alexander Solomo	0	0	170	300	360
andreas Vilholm V	0	0	170	300	360
Ahmad Shereef	0	0	170	300	360
Chenxi Cai	0	0	170	300	360
Isabel Grimmig Ja	0	0	170	300	360
Oliver Fielder	0	0	170	300	360

Kilde- og litteratur liste.

- lans slides fra lektion 2, side 118, Udviklingsmetoder og it-systemer (30/10-2020)
- http://cm-consult.com/it-ordbog/junit/ (30/10-2020)
- CDIO del1 (30/10-2020)
- https://www.w3schools.com/java/java_classes.asp (28/10-2020)
- http://www.computerdk.com/Networking/other-computer-networking/79051.html (29/10-2020)
- https://da.bccrwp.org/compare/difference-between-sequence-diagram-and-collaboration-diagram/ (29/10-2020)
- https://www.tutorialspoint.com/junit/junit_test_framework.htm?fbclid=IwAR3pTIc9y0FYMjSyRGirQZ9mB2FohrvJdFs1Tqg4yVaIKIu1C_Qs_Tqex6w (30/10-2020)
- https://www.w3schools.com/?fbclid=IwAR1HxSlP6Xj0AiCe6Z_n3Phd4pcjBgYtMZtUto-SlMJEhadjYcZHYYRqqcg (30/10-2020)
- https://github.com/chen0046/22_del2