DANMARKS TEKNISKE UNIVERSITET

CDIO_2

02312-13-14-15

ADAM KOSAK (s175121)
AHAD IMTIAZ (s175128)
CHRISTIAN RIDDERSHOLM HØJ (s175125)
CHRISTOFFER VOIGT (s154308)
GUNN MOHR HENTZE (s145494)
JANUS OLSEN (s175129)

Timeregnskab

Deltager	Analyse	Design	Implemente-ring	Test	Diverse	I alt
Gunn	2	2	10	2	2	18
Ahad	4	4	4	2	3	17
Adam	2	5	2	1	6	16
Janus	2	5	10	1	3	21
Christoffer	4	4	2	1	3	14
Christian	4	3	2	2	3	15
I alt	18	18	30	9	20	95

Indholdsfortegnelse

1. Indledning & Problemformulering

1.1 Indledning

I dette projekt udvikles et spil, som kan installeres og anvendes på DTUs databarer. Opgaven er blevet udarbejdet ved hjælp af undervisningsfagene, "Udviklingsmetoder til IT-systemer (02313)", "Indledende programmering (02314)" og "Versionsstyring og testmetoder (02315) ". Løsningsforslaget udarbejdes derfor i overensstemmelse med fagenes metoder. Rapportens formål er at dokumentere arbejdsprocessen for virksomheden IOOuterActive under udviklingen af spillet. Desuden vil der blive udført forskellige test for at sikre, at spillet fungerer hensigtsmæssigt.

1.2 Problemforumlering

Kunden, Danmarks Tekniske Universitet, ønsker et spil for to personer. Visionen fra kundens side er, at der skiftevis kastes med to terninger. De mulige øjental (2-12) repræsenterer hver især et felt, som indeholder et navn, en kort beskrivelse og en økonomisk konsekvens. Hver spiller modtager en startbeholdning på 1000. Spillerne kaster terningerne indtil en af dem opnår en beholdning på 3000 eller derover, og dermed vinder vedkommende spillet. Derudover ønsker kunden et Graphic User Interface, hvis IOOuterActive har tid implementere den.

2. Analyse

2.1 Use Case Beskrivelser

Brief

To spillere sætter sig ved samme computer og spiller et spil. De to spillere slår skiftevis med to terninger om først at nå en pengebeholdning på 3000. Spillerne starter med en pengebeholdning på 1000, og der gælder forskellige regler for alle de mulige slag (2-12), der alle har positiv eller negativ effekt på spillernes pengebeholdning.

Casual

To spillere sætter sig ved samme computer og spiller et spil. De to spillere slår skiftevis med to terninger og lander på et felt med numre fra 2-12. At lande på et af disse felter kan have en positiv eller en negativ effekt på spillernes samlede pengebeholdning. Effekten af de forskellige felter ses herunder:

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

Spillerne starter med en pengebeholdning på 1000 og den første til at nå en pengebeholdning på 3000 eller over vinder.

Fully Dressed

Use case navn	Spil
Use Case Nr.	1
Anvendelsesområde	Underholdning
Primære Aktør	Spiller 1 & Spiller 2
Interessenter	Danmarks Tekniske Universitet • Vil have et spil på computererne på deres databarer. IOOuterActive • Vil lave et velfungerende spil til underholdning, som bliver installeret på DTUs databarer og kører uden bemærkelsesværdige forsinkelser. Brugeren • Vil have et simpelt underholdende spil, som kan spilles mens vedkommende befinder sig i DTUs databarer.
Forhåndsbetingelser	• En velfungerende mus eller touchpad koblet til en computer. • Spillet er startet op
Succes Garanti	To spillere kaster skiftevis med to terninger. Terningernes værdi afgør spillernes bilers placering på felterne. Felterne øger eller mindsker spillernes pengebeholdning og pengebeholdningen kan aldrig blive negativ. Første spiller som opnår en pengebeholdning på 3000 eller mere har vundet spillet.
Main Flow	Spillet påbegyndes. Spillerne indtaster deres navne og Hver spiller starter med en pengebeholdning på 1000. To spillere kaster skiftevis et par af terninger. Terningernes værdier bliver vist når spilleren har kastet dem.

	Spillerens bil rykkes til et felt efter terningernes samlede værdi. Spillerens pengebeholdning vil henholdsvis øges eller mindskes alt efter feltets funktion. Spilleren vinder spillet når pengebeholdningen er 3000 eller over, spillet afsluttes.
Alternative Flow	Tekst ved felter 1. - 2. Du har besejret en ond troldmand i tårnet finder hans skat på 250 3. Du er faldet ned i krateret på en vulkan og må betale 100 for at blive reddet op. 4. Du har bygget nogle flotte palads port og kejseren belønner dig med 100 5. Grundet et opgør med den lokale høvding er du nødt til at flygte gennem kolde ørken og må betale 20 for rationer til turen. 6. Som leder af den bemurede by får du 180 i skatteindtægter. 7. Munkene i klostret beskytter dig mod onde fjender og onde ånder, til gengælde tjener du ingen penge. 8. Foran den sorte hule løber alle dine lejesoldater væk i frygt og du mister 70 9. Du finder en gruppe hytter i bjergene med et imødekommende folk som sælger dig ædelsten. Modtag 60. 10. På din rejse mod nord møder du en mur befæstet med varulve. I din flugt møder du en handelskaravane. Du betaler dem 80 for hurtigt at komme væk. Slå igen. 11. Du falder ned i et dybt hul og mister 50 i dine strabadser for at komme ud. 12. Under en udgravning efter fossiler finder du en guldmine! Du får 650!
Specielle krav	-
Teknologi og Data variantionsliste	• Forskellige versioner af Windows • Forskellige versioner af Java
Frekvens af forekomster	Hver gang programmet anvendes.
Diverse	-

2.2 Supplerende specifikationer (FURPS+)

Supplerende specifikationer indebærer andre krav, informationer og begrænsninger, der ikke umiddelbart beskrives i Use Cases. Specielt de ikke-funktionelle krav skal beskrives i de supplerende specifikationer.

Functionality

- Sikkerhed: Der er ingen fortrolig data, der skal tages højde for.
- Genanvendelse: Programmet skal kunne installeres på flere af databarens computere.

Usability

- Brugervenlighed: Programmet er udviklet således at brugervenligheden er høj. Der kræves meget lidt fra brugerens side ved brug af programmet.
- Hjælp: En passende brugervejledning er udarbejdet, til både installation af programmet samt til spillets gang.
- Dokumentation: Der medfølger en installationsvejledning.

Reliability

- Robusthed: Systemet skal være modstandsdygtigt overfor brugerfejl.

Performance

- Svartider: Det kræves af programmet, at der som udgangspunkt ikke må gå mere end ⅓ af et sekund, før programmet svarer på brugerens kommando. (Der ses her bort fra programmets første kørsel (koldstart).

Supportability

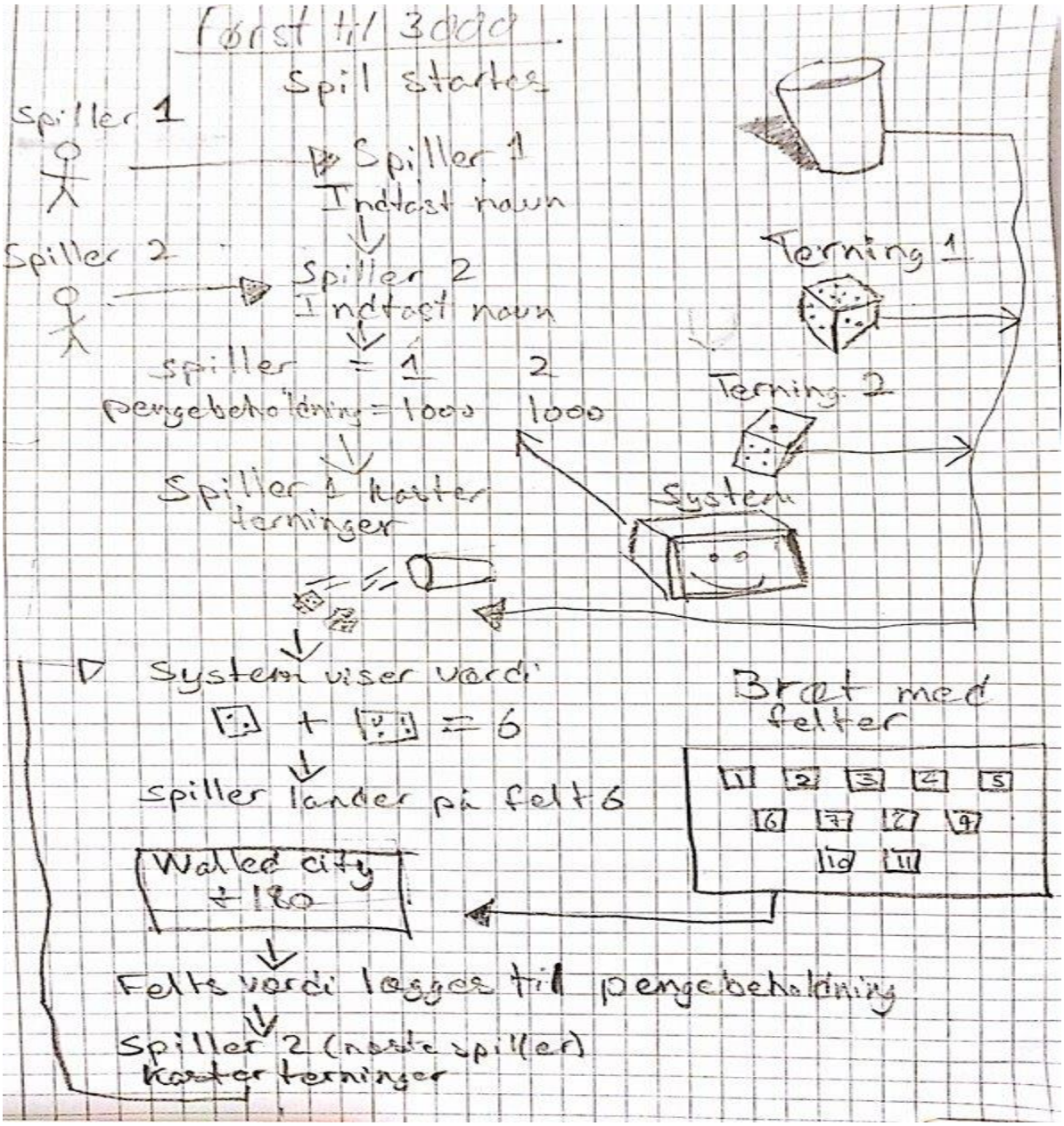
- Vedligeholdelse: IOOuterActive vil stå for vedligeholdelsen, hvis der skulle opstå problemer med systemet.

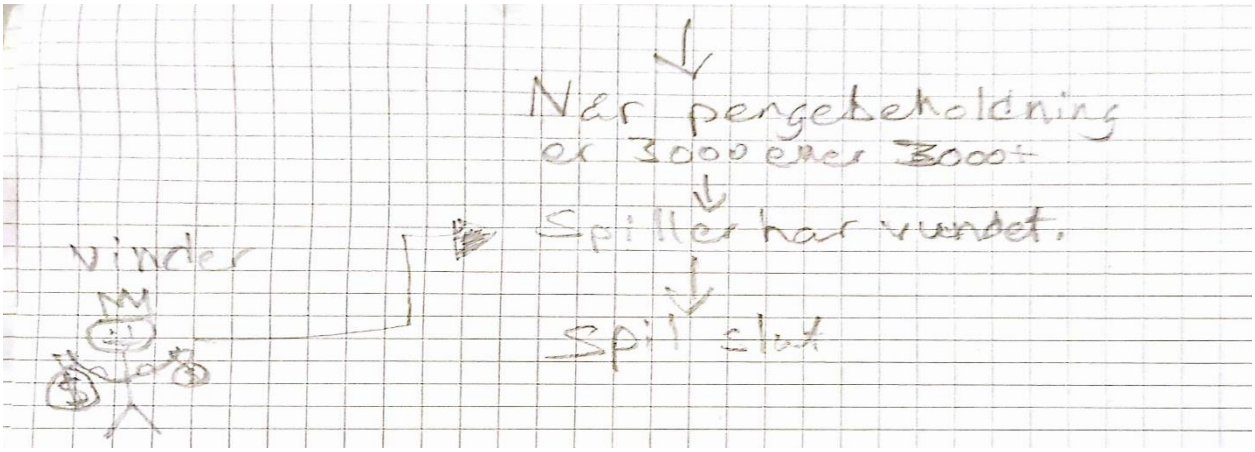
Implementation (+)

- Spillet udarbejdes i programmeringssproget Java.
- Spillet skal kunne køre på maskinerne på DTUs databarer, dvs. på Windows med tilhørende java plug-in.

2.3 Rige billeder

Visuel beskrivelse af hvordan vi opfatter spillets gang.

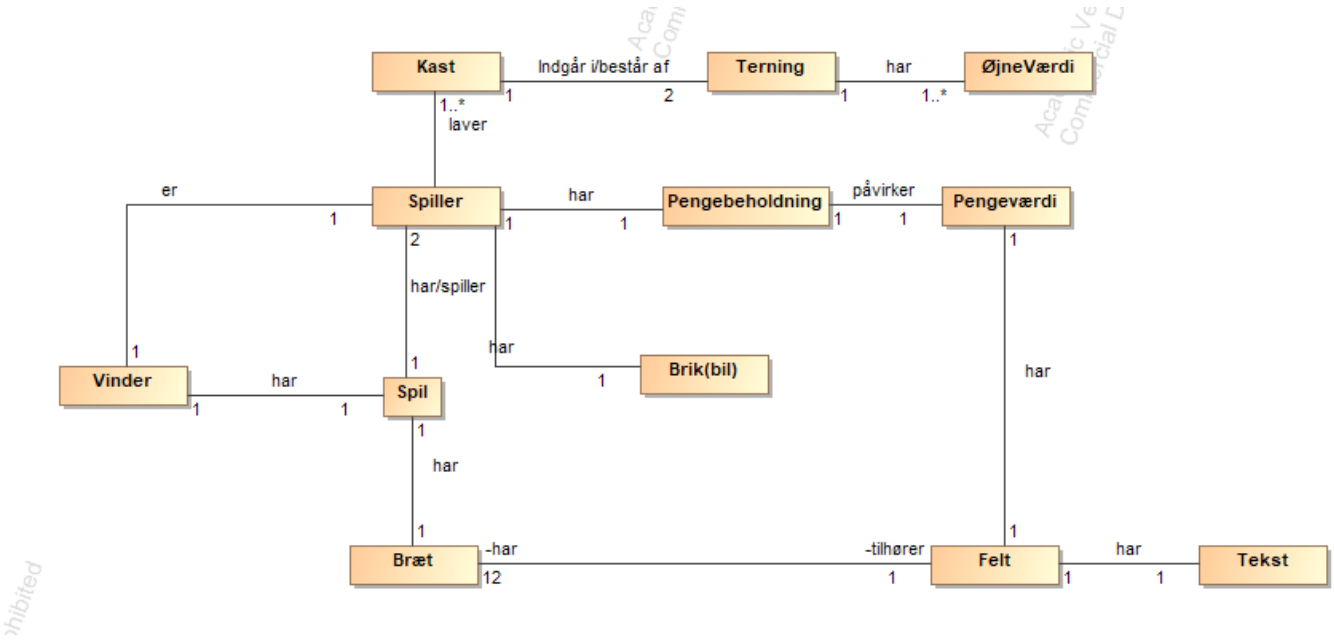




Figur 2.1: Rigt billede, som har til formål at illustere spillet.

2.4 Domænemodel

Moddellereing af virkeligheden som den så ud før udviklingen af programmet gik igang. Der er i domænemodelen forsøgt at afklare de vigtigste klasser, og spare på attributter og metoder.



Figur 2.2: Spillets domænemodel

2.6 Risikoanalyse

For at udarbejdelsen af projektet foregår problemfrit, opstilles de mulige risici, hvorefter disse vurderes og organiseres ud fra et risikoindeks. Projektgruppen er nu opmærksom på de mulige

risici, og deres indtræffelse kan derfor delvist forebygges. Risci med højest indeks håndteres naturligvis først. Det skal dog bemærkes, at visse risici ikke kan forebygges.

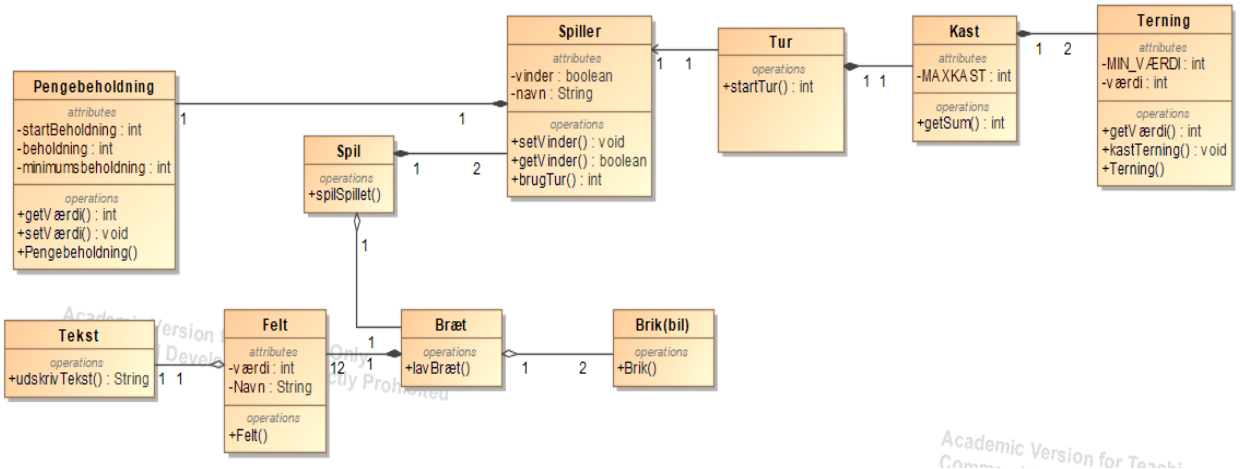
Risici	Sandsynlighed (x)	Skadevirkning (y)	Risikoindeks (x*y)
Misforståelse af projektkravene.	0.2	9	1.8
Spillets elementer fungerer ikke hensigtsmæssigt.	0.2	7	1.4
Dele af projektet bliver afleveret for sent af et eller flere gruppemedlem/ gruppemedelemmer.	0.4	3	1.2
Graphic User Interface fungerer ikke hensigtsmæssigt.	0.2	5	1
Der forekommer ændringer i projektkravene.	0.1	8	0.8
Der opstår en konflikt blandt gruppemedlemmerne.	0.2	3	0.6
Et gruppemedlem forlader projektet.	0.1	5	0.5
Hardware fejl på en af gruppens computere.	0.1	3	0.3

Figur 2.3: Risikotabel

3. Design

3.1 Design – klassediagram

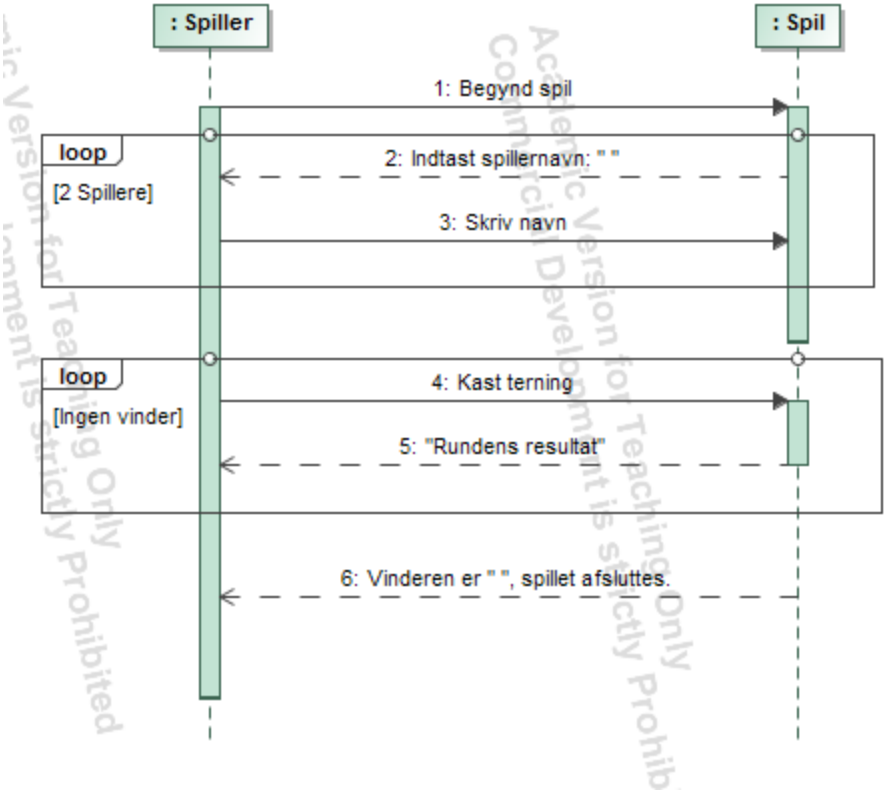
Alle navne i designklassediagram er i overenstemmelse med koden, da den bruges som udgangspunkt i programmeringsarbejdet.



Figur 3.1: Spillets Design - klassediagram

3.2 System-sekvensdiagram

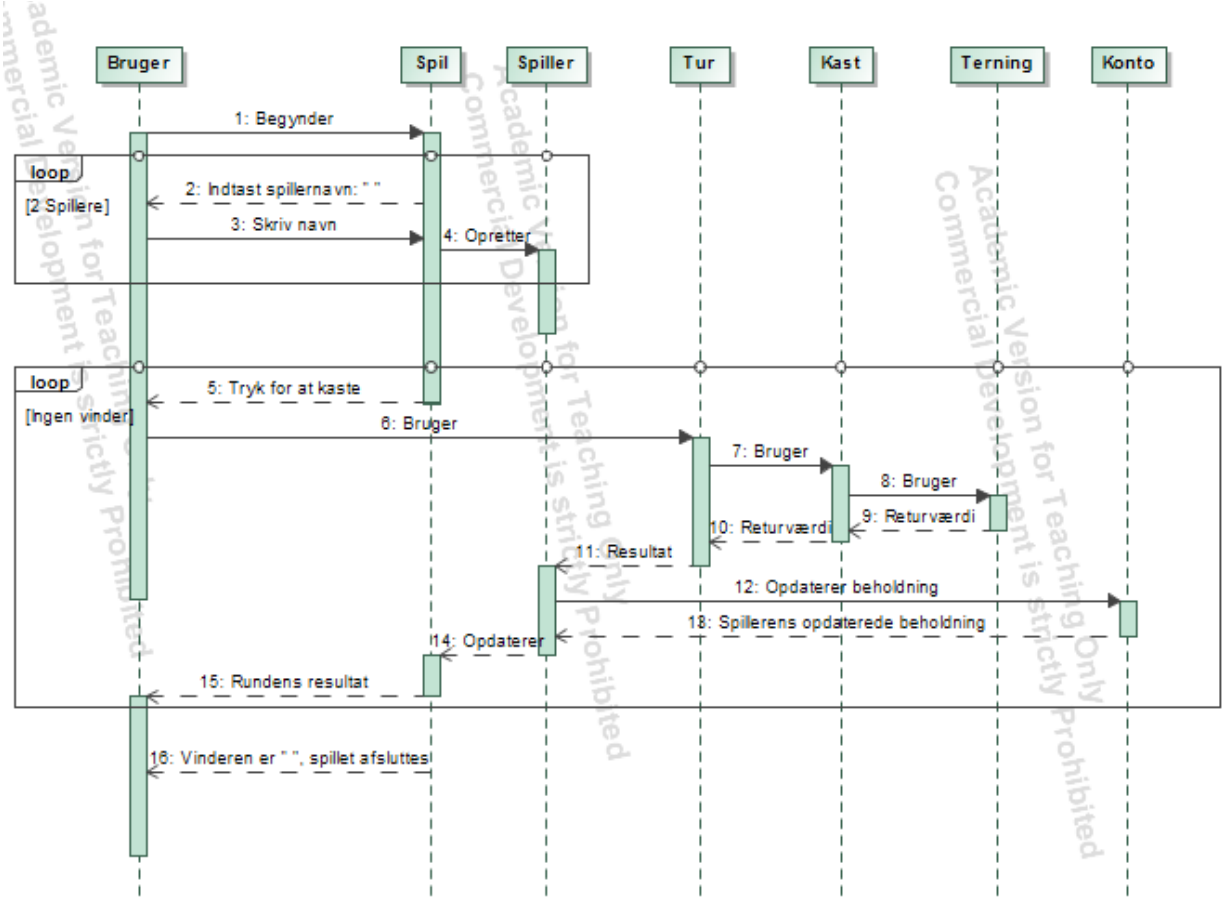
System-sekvensdiagrammet viser interaktionen mellem aktører og system under Use Casen ”Spil”. Diagrammet viser samme information som den tilsvarende Use Case beskrivelse, og er sproguafhængigt.



Figur 3.2: Spillets System - sekvensdiagram

3.3 Sekvensdiagram

Sekvensdiagrammet viser hvordan de vigtigste klasser og objekter interagerer med hinanden, når programmet gennemløbes.



Figur 3.3: Spillets Sekvensdiagram

4. Implementering

4.1 GRASP

Grasp (*General Responsibility Assignment Software Patterns*) består af metoder til fordeling af ansvar mellem objekter og klasser, indenfor objekt orienteret programmering.
Spillet er udarbejdet efter udvalgte GRASP principper, til at fordele ansvar og opgaver blandt programmets forskellige klasser.
Det udarbejdede designklassediagram viser resultatet af ansvarsfordelingen.

Creator

Creator mønsteret giver en beskrivelse til, hvor det vil være optimalt at instantiere objekter. Under dette princip betragtes, hvad der bruger objektet, hvad der har data til objektet, hvad der bliver gemt i objektet og om dette objekt er del af en komposition under et andet objekt. Kort

sagt vil det sige at, hvis klasse A har ansvaret for objektet af klasse B, vil det være A's ansvar at instantiere B. I projektet er der blandt andet *Pengebeholdning*-klassen som instantieres når der oprettes en spiller fra *Spiller*-klassen.

Information Expert

*Information Exper*t-princippet går ud på, som det ligger i navnet, at klassen som er ansvarlig for informationen, skal indeholde informationen.
Princippet hjælper med at uddelegere ansvaret for beregninger og metoder til de klasser hvor de hører til. I dette projekt er et eksempel klassen *Pengebeholdning*. Klassen skal angive en spillers pengebeholdning, hvormed den også har metoder som forøger eller mindsker pengebeholdningens værdi, samt angiver spillerens samlede pengebeholdning. Dette princip er der lagt meget vægt på i projektet til udarbejdelse af de forskellige klasser og deres metoder.

Controller

Her vil det sige at man blot skal have én klasse, som tager sig af at kontrollere de enkelte klassers sammenhæng for hvert use case. Dette bidrager til at en lettere forståelig kode som også nemt kan redigeres. I dette projekts tilfælde findes ét use case, da det er et simpelt spil, hvormed klassen '*Spil*' kontrollere spillet.

Low Coupling

Low Coupling princippet sikrer at så lav afhængighed mellem klasser som muligt, så påvirkningen af andre komponenter ikke sker ved ændringen i en en klasse, hvilket medfører en naturlig fleksibilitet ved udviklingen af programmet. Når et element ikke er afhængig af eller forbundet med for mange andre elementer, har det lav kobling. Ved udviklingen af programmet er den lave kobling eftertænkt gennem hele processen, og udført efter bedste evne. Den lave kobling mellem klasserne gør det let at forstå de forskellige dele, og er med til at gøre programmet let genbrugeligt.

High Cohesion

High Cohesion er med til at støtte den lave kobling i programmet, og bruges i programmet til at bevare objekterne fokuserede og forståelige. Programmet bliver også nemmere at genbruge til andre projekter. Ved High Cohesion kigges der på klasserne i forhold til ansvarsområde, om de har meget eller lidt ansvar. Programmet er udviklet efter tanken om, at ansvarsområdet for den enkelte klasse ikke må blive for stort. Dette er gjort gennem opdeling af klasser, så der fås flere klasser med mindre ansvar hver især. Opdelingen af klasser i programmet ses blandt andet ved klasserne bræt og felt, som kunne samles i én klasse, bræt, men dette ville giver bræt, et for stort ansvarsområde.

4.2 Beskrivelse af klasser og objekter

Kast
I klassen Kast, er der udviklet metoder til oprettelse af terninger, samt metoder som samler summen af værdierne af terningerne som fås fra *Terning*-klassen.

Terning
Terning-klassen anvendes til at angive en værdi for en terning. Klassen er kodet således at terninger til spillet kan justeres til at angive forskellige minimum og maksimum værdier. F.eks. kan der i stedet for en sekssidet terning anvendes en 11 sidet terning (2-12).

Tur
Klassen anvendes til at starte og skifte en spillers tur. Klassen har metoder til at hente summen af værdierne fra terningkastet. Den indeholder også metoder til at bestemme hvilket felt man lander på, alt efter terningers sum, samt felternes tilhørende værdi.

Spiller
Spiller klassen anvendes til at oprette og gemme en spillers navn og tilknytte en tur til spilleren. Det er også i denne klasse metoden for at angive vinderen af spillet er oprettet. Der er yderligere metoder for at tilkoble en startbeholdning til hver spiller oprettet.

Konto
Klassen *Konto* illustrerer spillerens samlede pengebeholdning. Klassen indeholder metoder der forbinder og påvirkes af felternes tilhørende værdi, fra *Bræt*-klassen, som henholdsvis øger eller mindsker spillerens pengebeholdning. *Konto*-klassen er sat til at have en startværdi på 1000, samt en minimumsgrænse, så værdien ikke går under 0. Klassen har også metoder, som bestemmer når en spiller opnår en beholdning, som kvalificeres til at vinde spillet.

Sprog
Klassen sprog indeholder spillets tekster, som skal vises ved felterne, når man startet spillet og når man har vundet spillet.

Bræt
Denne klasse bruges til at danne felterne hvor spillerens brik kan lande på, alt efter værdien af terningkastets sum. *Bræt*-klassen indeholder tekst og billede for hvert felt på spillebrættet samt en tilhørende værdi, som påvirker spillerens pengebeholdning. Dertil er der en metode som tilknytter spillereglerne til brættet.

5. Test

Et program kan have en del fejl, store som små, som er overset under produktionen eller ikke har været en fejl i producenternes øjne. Derfor er der gennemført en række test under selve udarbejdelsen, for at eliminere de væsentlige fejl i programmet, før den udleveres til kunden. Samtidig er der også udarbejdet en brugertest, udført af en person uden kendskab til projektet, for at teste brugervenligheden af spillet og for at finde fejl, som ikke blev bemærket af IOOuterActive. Testene har til formål at forbedre spillets kvalitet, således at brugeren får en god oplevelse samt, at virksomheden slipper for klager fra kunden.

5.1 Junit test

De udførte Junit test er inkulderet i programmet.

Test ID	TC01
Referat	Automatiseret test af opdaterBeholdning
Krav	Beholdningen må aldrig blive negativ.
Betingelser før	Beholdningens værdi er 0. Minimumsbeholdningen er 0.
Betingelser efter	Beholdningens værdi er 0.
Testens fremgangsmåde	1. Der oprettes en Junit test case. 2. Der opstilles en passende testmetode. 3. Junit test casen køres i Eclipse.
Test data	-10
Forventet resultat	0
Faktisk resultat	0
Status	Bestået
Testet af	Gunn Mohr Hentze
Dato	10-11-2017
Testmiljø	Eclipse 4.6.2 på Windows 10

Test ID	TC02
Referat	Automatiseret test af opdaterBeholdning
Krav	Beholdningen mindsker
Betingelser før	Beholdningens værdi er 1000 Minimumsbeholdningen er 0.
Betingelser efter	Beholdningsværdien er opdateret
Testens fremgangsmåde	1. Der oprettes en Junit test case. 2. Der opstilles en passende testmetode. 3. Junit test casen køres i Eclipse.
Test data	-180
Forventet resultat	820
Faktisk resultat	820
Status	Bestået
Testet af	Gunn Mohr Hentze
Dato	10-11-2017
Testmiljø	Eclipse 4.6.2 på Windows 10

Test ID	TC03
Referat	Automatiseret test af opdaterBeholdning
Krav	Beholdningen øges
Betingelser før	Beholdningens værdi er 1000. Minimumsbeholdningen er 0.
Betingelser efter	Beholdningsværdien er opdateret
Testens fremgangsmåde	1. Der oprettes en Junit test case. 2. Der opstilles en passende testmetode. 3. Junit test casen køres i Eclipse.
Test data	180
Forventet resultat	1180
Faktisk resultat	1180
Status	Bestået
Testet af	Gunn Mohr Hentze
Dato	10-11-2017
Testmiljø	Eclipse 4.6.2 på Windows 10

Test ID	TC04
Referat	Automatiseret test af opdaterBeholdning
Krav	At der findes en vinder
Betingelser før	Beholdningens værdi er 2990. Minimumsbeholdningen er 0.
Betingelser efter	Beholdningens værdi er 3000
Testens fremgangsmåde	1. Der oprettes en Junit test case. 2. Der opstilles en passende testmetode. 3. Junit test casen køres i Eclipse.
Test data	1000
Forventet resultat	True
Faktisk resultat	True
Status	Bestået
Testet af	Gunn Mohr Hentze
Dato	10-11-2017
Testmiljø	Eclipse 4.6.2 på Windows 10

6.1 Installationsvejledning

Følgende brugertests er udført på det færdige spil. Nedenfor er en oversigt over stillede spørgsmål i testen.

Spørgsmål	**Svar**
• Forstår du hvad spillet går ud på, uden producenternes hjælp?	
• Har du brug for en brugervejledning for at spille spillet?	
• Er spillets tekster for at gennemføre spillet forståelige?	
• Er spillet brugervenligt?	
• Tog det for lang tid at spille spillet færdigt?	

• Har du evt. forbedringer til spillet?	

Brugertesten er udført af en person uden kendskab til projektet eller indmaden af projektet. Testen er udført af brugeren uden interaktion fra projektets deltagere, dvs. Testbrugeren har været i kontrol under hele testen, og er blevet observeret af en eller flere af projektets deltagere. Medfølgende noter er skrevet ud fra brugerens tanker, kommentarer og reaktioner ved test af programmet.

Bruger test 1
Test-Dato: 09/11-2017, kl. 22:45

Spørgsmål	Svar
• Forstår du hvad spillet går ud på, uden producenternes hjælp?	Ja
• Har du brug for en brugervejledning for at spille spillet?	Nej
• Er spillets tekster for at gennemføre spillet forståelige?	Ja
• Er spillet brugervenligt?	Nej
• Tog det for lang tid at spille spillet færdigt?	Nej
• Har du evt. forbedringer til spillet?	En bedre afslutning

Yderligere bemærkninger fra testpersonen:

- Troede bilen skulle rykke sig i stedet for bare at hoppe → Bedre forklaring
- Point ikke synlige nok → Større skrift
- Det er et dårligt spil - kedeligt
- Forviring ved afslutningen
- Tydeligt ved starten at navne skulle skrives ind
- Gode kort der var på
- Det er forvirrende hvad man skal
- Tvivl om man skal lave andet end bare at trykke
- Virker ikke som om man kommer videre

Ved gennemløb af programmet har IOOuterActive selv noteret følgende:

- Spillet stopper meget brat når man vinder, man mister muligheden for at slå igen, og der står ikke f.eks. “Christian du har vundet”

- Felt 6 mangler tekst.
- Måske lidt mere information på startskærmen inden spillet begynder.
- Når man lander på et felt kommer der ingen tekst frem, kun “Prøv lykken”.

6. Brugervejledning

6.1 Installationsvejledning

En vejledning til at køre programmet:

1) Installer Java
2) Installer spil-filen
    a) Hent det fra IOOuterActives hjemmeside
3) Spillet startes op og er klar til at køre
    a) Programmet laves som en runnable Jar-fil så det fungerer fint på Windows systemer.

7. Konklusion

7.1 Konklusion

IOOuterActive har under sit andet projekt fremstillet et tilfredstillende produkt samt dertilhørende dokumentation, i overenstemmelse med opgavekravene udarbejdet i undervisningsfagene “Udviklingsmetoder til IT-systemer (02313)”, “Indledende programmering (02314)” og “Versionsstyring og testmetoder (02315) ”. Desuden har projektgruppen tilkoblet spillet en Graphic User Interface. Spillet har efterfølgende været udsat for en række test, som har sikret at det virker hensigtsmæssigt.