DANMARKS TEKNISKE UNIVERSITET

CDIO_3

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	Implementering	Test	Dokumentation	Diverse	I alt
Adam	5	9	11	1	4	1	31
Ahad	6	3	5	5	8,5	1	28,5
Christian	1,5	8	6,5	7,5	4	3	30,5
Christoffer	5	6	2	2	4	3	22
Gunn	9,5	3	31,5	1	2	2	49
Janus	5	6	16,5	2	1	1	31,5
I alt	32	35	72,5	18,5	23,5	11	192,5

Tabel 1: Oversigt over timeantal, hver gruppemedlem har brugt på de forskellige dele af projektet

For udvidet timeregnskab, se bilag1

1 Indledning og 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 Problemformulering

Kunden, Danmarks Tekniske Universitet, ønsker et spil for to til fire personer. Visionen fra kundens side er, at der skiftevis kastes med en terning. Ud fra det viste øjnetal, flytter spillerne sig rundt på spillerpladen blandt ejerfelter, chancefelter og diverse. Hver spiller modtager en startbeholdning, hvis størrelse afhænger af antallet af spillere. Spillerne kaster terningen indtil en af dem ikke har penge til at købe en ejendom, eller betale en husleje. Spilleren med den største pengebeholdning på dette tidspunkt, er vinderen af spillet.
Derudover ønsker kunden et Graphic User Interface, hvis IOOuterActive har tid implementere den.

2 Analyse

2.1 Funktionelle krav

Must have:

1. Der skal være en spilleplade.
2. Spillepladen skal have forskellige typer af felter.
3. Spiller-antallet skal kunne variere mellem 2-4 spillere.
4. Hver spiller skal have en brik.
5. Spillernes brikker skal kunne blive stående på felter de lander på, og fortsætte videre derfra
6. Spillernes brikker skal kunne gå i ring på spillepladen
7. Der skal udvikles en brugergrænseflade til spillet
8. Der skal være 24 felter på pladen
9. Startbeholdningen for to spillere skal være 20M hver.
10. Startbeholdningen for tre spillere skal være 18M hver.
11. Startbeholdningen for fire spillere skal være 16M hver.
12. Der er kun én terning i spiller.
13. Spillet skal have 20 chancekort.
14. Alle spillere starter på feltet "start".
15. En spiller skal rykke det antal frem, som de slår med terningeslaget
16. Hver gang en spiller passerer eller lander på "Start-feltet, modtager spilleren 2M.
17. Lander en spiller på et ledigt felt, SKAL den købes af spilleren
18. Lander en spiller på et felt, ejet af en anden spiller, betales det felt-bestemte lejebeløb til ejeren.
19. Ejes begge felt i samme farve af samme spiller, er huslejen dobbelt så høj.
20. Lander en spiller på et "chancen-felt, trækkes et chancekort. Spilleren følger en anvisning.
21. Landen en spiller på "gå i fængsel-feltet, rykker spillerens brik til "på besøg i fængsel-feltet. Selvom spillere passerer start, modtager denne ikke 2M.
22. Er en spiller "fanget"i fængsel, skal spilleren i næste tur betale 1M, eller bruge chancekortet "du løslades uden omkostninger".
23. En spiller fanget i fængsel kan sagtens opkræve husleje, når en anden spiller lander på et felt, ejet af spilleren.
24. Lander en spiller på "på besøg-feltet, bliver spillerens brik stående til næste omgang.
25. Lander en spiller på "gratis parkering-feltet, bliver spillerens brik stående til næste omgang.
26. En spiller går fallit, hvis spilleren ikke er i stand til at betale husleje
27. En spiller går fallit, hvis spilleren ikke er i stand til at købe en ejendom
28. En spiller går fallit, hvis spilleren ikke er i stand til at betale en afgift
29. Den spiller, der har flest penge på det tidspunkt, hvor en anden spiller er fallit, vinder spillet.

30. Har to eller tre spillere lige mange penge, når en anden spiller går fallit, tælles ejendommenes værdi for at afgøre, hvem der vinder.
31. Der skal anvendes et Git-repository

Should have:

33. Chancekortene skal blandes før start
34. Chancekort, der bruges, lægges nederst i bunken.
35. Fordelingen af chancekort skal følge det, i bilag 1 vedlagte materiale.
36. Ejendom-felterne kommer i par af to, som er farve bestemt.

Could have:

38. En spiller går fallit, hvis spilleren ikke er i stand til at sælge ejendomme til banken eller en anden spiller.
39. Kan en spiller ikke betale, hvad "banken"dikterer, skal spilleren betale gælden med en ejendom af samme værdi eller højere. Ejendommen bliver sat til salg igen af banken.
40. Kan en spiller ikke betale, hvad lejen hos en anden spiller dikterer, skal spilleren betale gælden med en ejendom af samme værdi eller højere.
41. Hver spiller vælger sin farve på brikken.

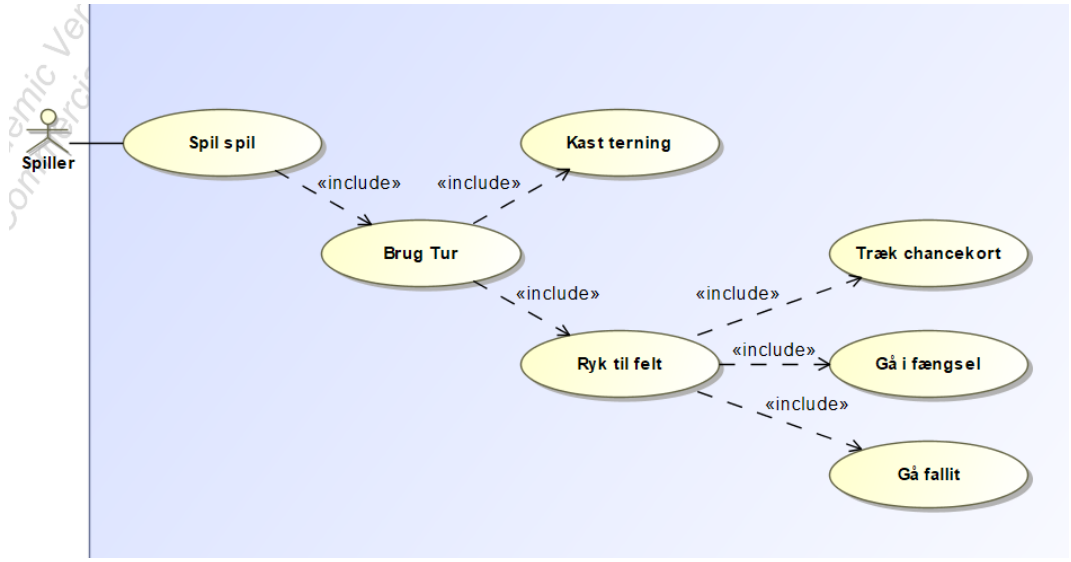
Want to have:

42. Den yngste spiller starter.

Felt	Navn	Type	M	Note
1	Start	Start		Får 2M hver gang en spiller passerer
2	Burgerbaren	Ejendom	1	Købe eller betale leje
3	Pizzariaet	Ejendom	1	Købe eller betale leje
4	Chance	Chance		Tager et chancekort og følger anvisningen
5	Slikbutikken	Ejendom	1	Købe eller betale leje
6	Iskiosken	Ejendom	1	Købe eller betale leje
7	På besøg	Fængsel		Lander på det: Sker ingenting I fængsel: Betale 1M eller chance kort for at komme ud
8	Museet	Ejendom	2	Købe eller betale leje
9	Biblioteket	Ejendom	2	Købe eller betale leje
10	Chance	Chance		Tager et chancekort og følger anvisningen
11	Skaterparken	Ejendom	2	Købe eller betale leje
12	Swimmingpool	Ejendom	2	Købe eller betale leje
13	Gratis parkering	Parkering		Sker ingenting
14	Spillehallen	Ejendom	3	Købe eller betale leje
15	Biografen	Ejendom	3	Købe eller betale leje
16	Chance	Chance		Tager et chancekort og følger anvisningen
17	Legetøjsbutikken	Ejendom	3	Købe eller betale leje
18	Dyrehaven	Ejendom	3	Købe eller betale leje
19	Gå i fængsel	Fængsel		Går til feltet "på besøg", og følger anvisningen "I fængsel"
20	Bowlinghallen	Ejendom	4	Købe eller betale leje
21	Zoo	Ejendom	4	Købe eller betale leje
22	Chance	Chance		Tager et chancekort og følger anvisningen
23	Vandlandet	Ejendom	5	Købe eller betale leje
24	Strandpromenaden	Ejendom	5	Købe eller betale leje

Tabel 2: Oversigt over spillepladens felter og deres funktioner

2.2 Use case diagram



Figur 1: Use case diagram

2.3 Use case Beskrivelser

Use-case: Spil

Brief
To til fire spillere sætter sig ved en computer og spiller et spil Monopoly Junior. Spillerne starter med en pengeholdning og skiftes til at kaste med en terning og rykke en brik rundt på en spilleplade. Spillepladen har forskellige felter, der har forskellige funktioner, som kan påvirke pengebeholdningen. Vinderen findes, når en spiller er gået fallit.

Casual
To til fire spillere sætter sig ved en computer og spiller et spil Monopoly Junior. Spillerne starter med en bengeholdning på 20, 18 eller 16 for hhv. to, tre eller fire spillere. Spillerne skiftes til at kaste med en terning og rykke en brik rundt på en spilleplade. Spillepladen har 24 felter: <

- Et start felt.
- Fire chance felt.
- Et besøg felt.
- Et gratis parkering felt.
- Et gå i fængsel felt.

Derudover er der 16 ejendomme, der hver har en pris, som er den samme købspris og husleje pris:

- Fire har en pris på 1.
- Fire har en pris på 2.
- Fire har en pris på 3.
- To har en pris på 4.
- To har en pris på 5.

Når en spillers pengebeholdning bliver mindre end 0, vinder den spiller, der har flest penge.

Use case navn	Spil
Use Case nr.	1
Anvendelsesområde	Underholdning
Primære Aktør	Spiller
Interessenter	Danmarks Tekniske Universitet: • Vil have et spil på computerne på deres databarer. IOOuterActive: • Vil lave et velfungerende spil til underholdning, som bliver installeret på DTUs databarer. • Skal køre uden bemærkelsesværdige forsinkelser. Brugerne: • Vil have et underholdende spil, som kan spilles mens vedkommende befinder sig i DTUs databar.
Forhåndsbetingelser	• En velfungerende mus eller touchpad koblet til en computer. • Spillet er startet op • Der skal være 2-4 spillere
Succes garanti	To til fire spillere kaster skiftevis med en terning, og rykker en brik rundt på en spilleplade. Hvert felt kan påvirke spillernes pengebeholdning. Når en spiller får en negativ pengebeholdning, vinder den spiller, som har højst pengebeholdning.
Main Flow	1. Spillerne oplyser, hvor mange de er, og indtaster deres navne efter alder, hvor den yngste er først. 2. include(Brug Tur)

Tabel 3

Use case navn	Brug Tur
Use Case nr.	2
Primære aktør	Spiller
Forhåndsbetingelser	• Spillet er startet op • Brugeren har oprettet sig som spiller
Main Flow	1. Hvis spiller er i fængsel (a) Hvis spiller har -fri fra fængsel' chance kort, bruges det for at komme ud gratis. (b) Ellers betales der 1M til banken. 2. include(Kast terninger) 3. include(Ryk til felt)
Slutbetingelser	Spiller er rykket til et felt og færdig med sin tur.
Alternativ flow	Fallit

Tabel 4

Use case navn	Kast terning
Use Case nr.	3
Primære Aktør	Spiller
Forhåndsbetingelser	Spiller er i gang med sin tur
Main Flow	1. Spiller kaster terning
Slutbetingelser	Systemet har fået en værdi af terningen.

Tabel 5

Use case navn	Ryk til felt
Use Case nr.	4
Primære Aktør	Spiller
Forhåndsbetingelser	• Spiller har kastet en terning • Systemet har læst værdien
Main Flow	1. Spillerens brik rykkes frem til et felt anvist af terningens værdi.
Slutbetingelser	Spillerens brik er rykket frem til et felt.
Alternative Flow	• Start-felt • Ejendom-felt • Chance-felt • På besøg-felt/Gratis parkering-felt • Gå i fængsel

Tabel 6

Use case navn	Start-felt - alternativ flow
Use Case nr.	4.1
Primære Aktør	Spiller
Forhåndsbetingelser	Spiller er landet på eller har passeret start
Main Flow	1. Spiller modtager 2M til sin pengebeholdning

Tabel 7

Use case navn	Ejendom-felt - alternativ flow
Use Case nr.	4.2
Primære Aktør	Spiller
Forhåndsbetingelser	Spiller er landet på et ejendom-felt
Main Flow	1. Hvis feltet er ledigt, skal spiller købe feltet 2. Ellers skal spiller betale husleje.
Alternative Flow	Fallit

Tabel 8

Use case navn	Chancefelt - alternativ flow
Use Case nr.	4.3
Primære Aktør	Spiller
Forhåndsbetingelser	Spiller er landet på chance-felt
Main Flow	1. Spiller tager et kort fra bunken. 2. Spiller følger kortets anvisning.
Alternative Flow	Fallit

Tabel 9

Use case navn	På besøg/Gratis parkering - alternativ flow
Use Case nr.	4.4
Primære Aktør	Spiller
Forhåndsbetingelser	Spiller er landet på På besøg-felt
Main Flow	1. Der sker ikke noget

Tabel 10

Use case navn	Gå i fængsel - alternativ flow
Use Case nr.	4.5
Primære Aktør	Spiller
Forhåndsbetingelser	Spiller er landet på Gå i fængsel-felt
Main Flow	1. Spillerens brik rykkes direkte i fængsel

Tabel 11

Use case navn	Fallit - alternativ flow
Use Case nr.	5
Primære Aktør	Spiller
Forhåndsbetingelser	Spiller skal betale penge
Main Flow	1. Spilleren har ikke nok penge til at betale beløbet
Slutbetingelser	1. Resterende spillere tæller deres pengebeholdning. 2. Spiller med højst pengebeholdning vinder.

Tabel 12

2.4 Supplerende specifikationer ((F)URPS+)

Functionality

- Spillet skal kunne installeres og anvendes på DTUs databarer.

Usability

- Brugervenlighed: Programmet er lavet brugervenligt sådan at en bruger kun skal trykke på de knapper som spillet angiver, og ikke skal tænke på andet end at vælge mellem de fremstillede valgmuligheder.

Reliability

- Robusthed:

Performance

- Svartider: Vi ønsker at vores program har svar tid på $\frac{1}{3}$ sekund eller lavere fra at brugeren interagerer med systemet til der visuelt sker noget på skærmen.
- Nøjagtighed:
- Ydeevne:

Supportability

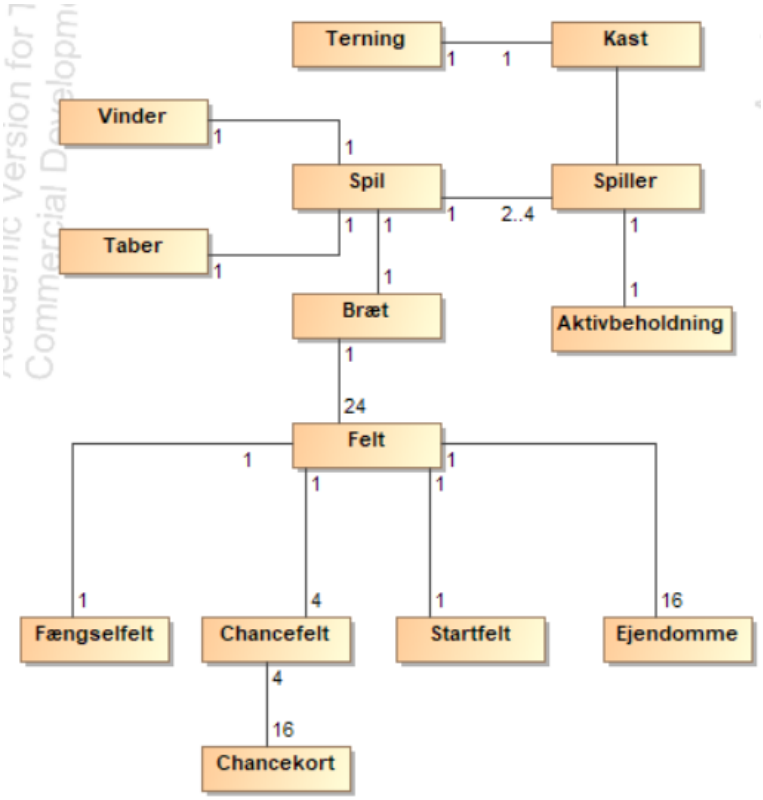
- Vedligeholdelse: Spillet bliver udviklet på en måde sådan at det efter udgivelse ikke har brug for yderligere support og kan fungere optimalt uden opdateringer.
- Anvendelighed: Spillet er lavet sådan at alle bare kan sætte sig ved en computer i databarene på DTU og spille uden nogen form for introduktion til hvordan de skal gøre.

Implementation

- Programmet udarbejdes i programmeringssproget Java

2.5 Domænemodel

Domænemodellen indeholder de vigtigste abstraktioner og informationer, der er nødvendige for at forstå domænet og de nuværende krav. De vigtigste klasser er afklaret. Desuden er navneordsanalyse anvendt på kravene for at udvælge navnene til henholdsvis klasser og metoder.



Figur 2: Domænemodel

2.6 Risikoanalyse

For at undgå så mange fejl som muligt, laves en liste over de mulige risici, som er involveret i udviklingen af projektet. Til de forskellige risici er der udarbejdet forslag til forebyggelse og håndtering, skulle risiciene indtræffe. Risiciene er farvekodet alt efter påvirkningskraft af projektet, så man ved, hvilke risici man skal fokusere på, bliver løst/forebygget. Det skal dog nævnes at ikke alle risici kan forebygges, men at man stadig klargøre løsninger skulle den pågældende risici indtræffe.

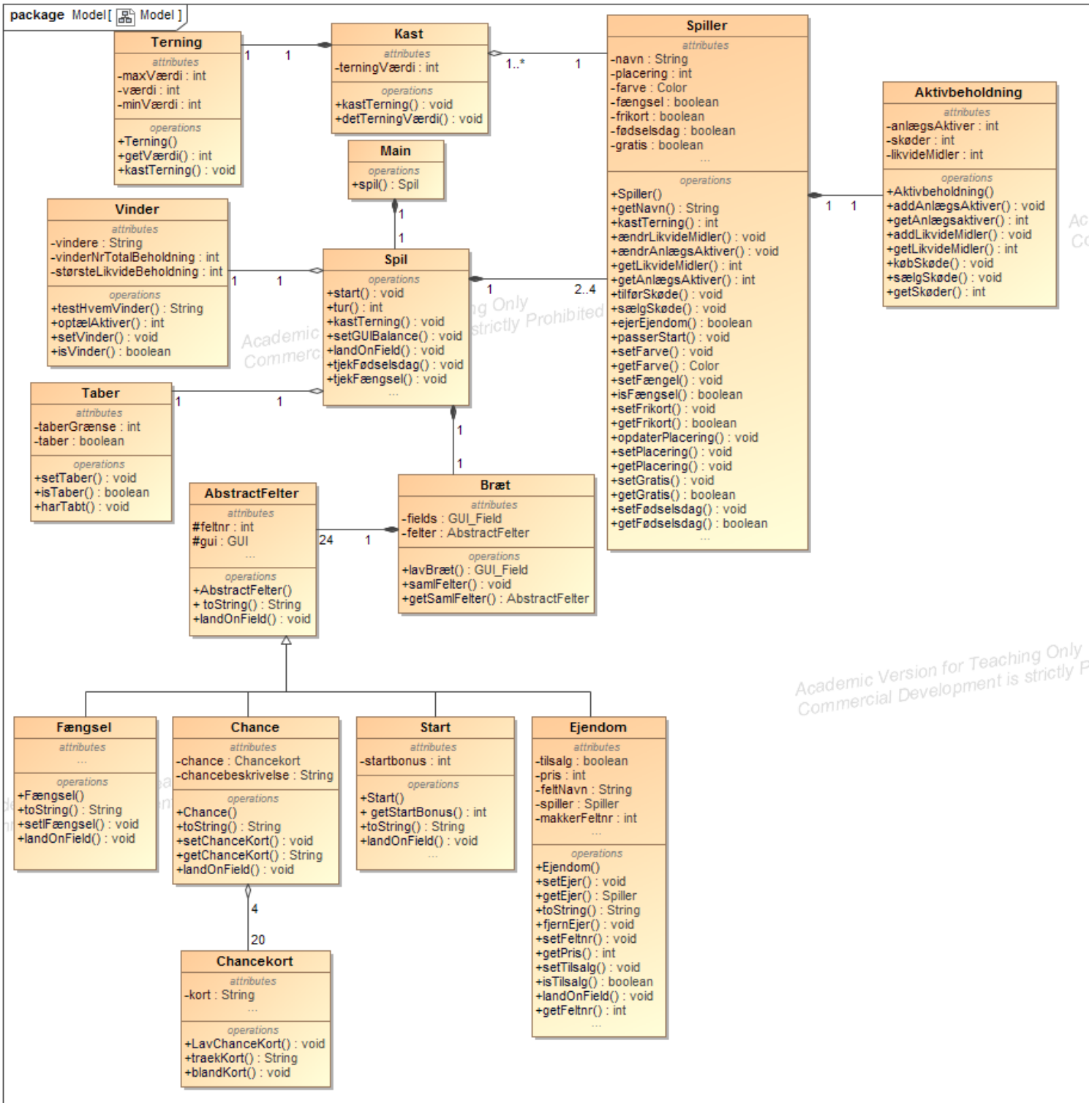
Risici	Sandsynlighed (0 <= x <= 1)	Skadevirkning (0 <= y <= 10)	Risiko indeks (x * y)	Forebyggelse og Håndtering
Spillets elementer fungere ikke hensigtsmæssig	0,2	7	1,4	Der gennemføres løbende test i Implen
Misforståelse af projektkravene	0,15	9	1,25	Forarbejdet til projektet er grundigt, projektgruppen skal have en fælles forståelse til projektkravene.
Dele af projektet bliver forsinket	0,4	3	1,2	Projektgruppen sørger for at opdatere h omkring deres arbejdsprocesser. Gruppemedlemmer støtter resten af pro hvis de er færdige med deres egen opg
Brugergrænsefladen fungerer ikke hensigtsmæssigt	0,2	5	1	Brugergrænsefladen implementeres tid Udarbejdelsen af programmet foregåro
Urealistiske tidsestimater	0,2	5	1	Sørge for, at forarbejdet er grundigt.
Der forekommer betydelige ændringer i projektkravene	0,1	8	0,8	Der holdes mødes med kunden tidligt i for at få en fælles vision for det færdig
Der forekommer mindre ændringer i projektkravene	0,2	4	0,8	Kontakt med kunden vedligeholdes un
Der opstår konflikter blandt gruppemedlemmerne	0,2	3	0,6	De ikke involverede parter fungerer so som har til opgave at løse konflikten fø
Et gruppemedlem forlader projektet.	0,1	5	0,5	Projektgruppen motiverer hinanden, og fordeler arbejdslasten rimeligt.
Hardware fejl på en af gruppemedlemmernes computer	0,1	5	0,5	Sørge for, at arbejdet jævnligt lægges o
Software fejl på en af gruppemedlemmernes computer	0,1	5	0,5	Sørge for, at arbejdet jævnligt lægges o

Tabel 13: My caption

3 Design

3.1 Design-klassediagram

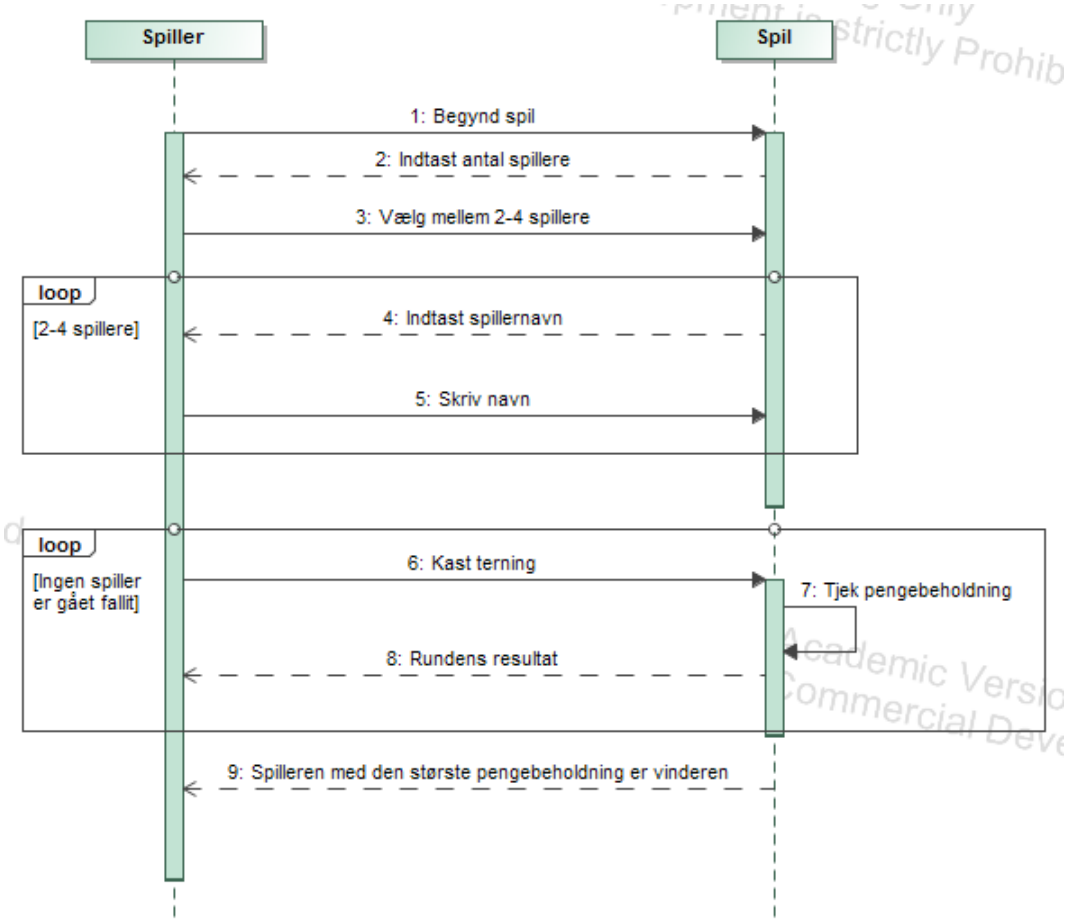
Design-klasse diagrammet er en visuel gengivelse programmet, med samtlige attributter og metoder angivet. Design-klassesiagrammet er i overensstemmelse med koden, da den bruges som udgangspunkt i programmeringsarbejdet. Der er undervejs i arbejdet med programmet lavet ændringer i design-klassediagrammet som følge af kodens udarbejdelse.



Figur 3: Design-klassediagram

3.2 Systemsekvensdiagram

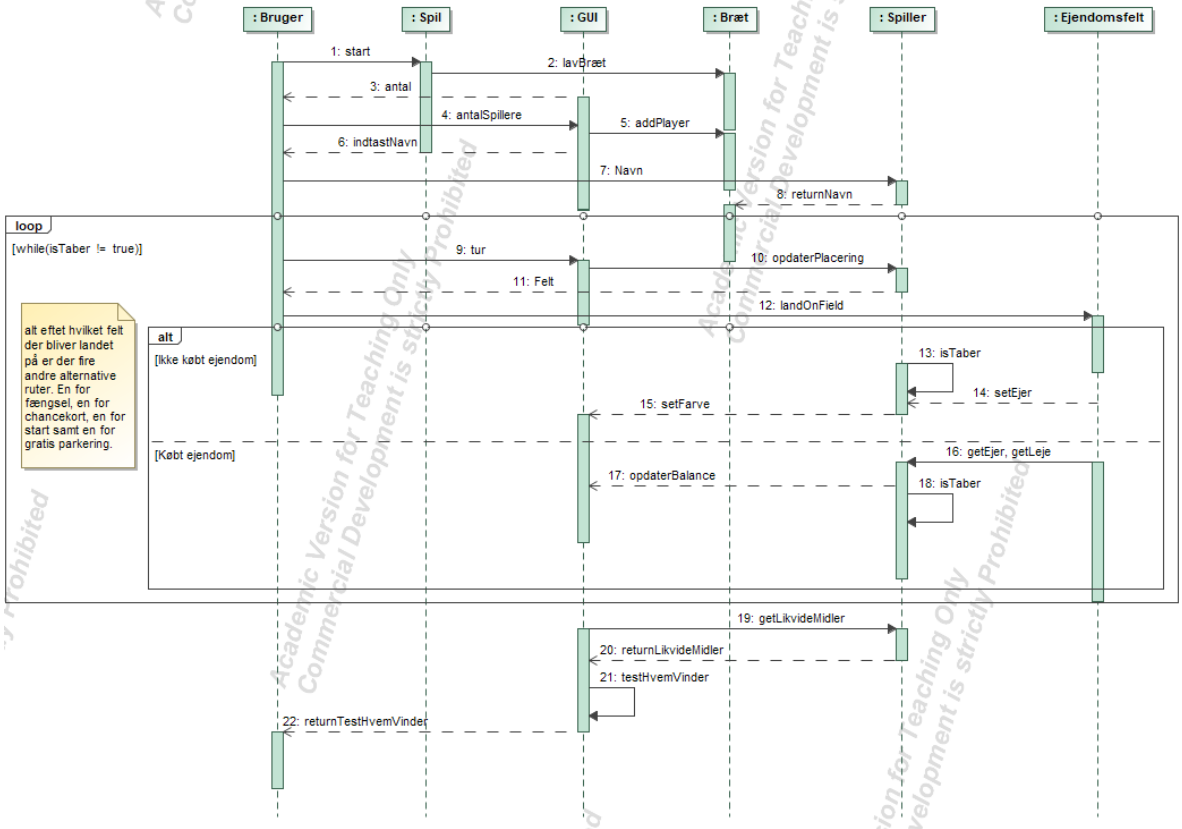
System-sekvensdiagrammet viser interaktionen mellem aktører og system under Use Casen "Spil". Diagrammet viser samme information som den tilsvarende Use Case beskrivelse, og er sproguafhængigt.



Figur 4: Systemsekvens diagram

3.3 Sekvensdiagram

Sekvens diagrammet viser hvordan vi forestiller os, at koden kommer til at fungere og interagere med de forskellige klasser og metoder. Diagrammet viser et spilforløb, hvor spillerne kun lander på ejendoms felter. Der findes fire alternative spil udfald, hvor de fire andre er, når man lander på felter af typerne: fængsel, chance kort, start(kan også bare passeres), samt gratis parkering. I sekvens diagrammet er nogle af klasserne som kommer til at blive brugt lagt ind under GUIen, da diagrammet ellers ville være blevet for uoverskueligt.



Figur 5: Sekvens diagram

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. Vinder klassen er i programmets tilfælde klasse B og Taber-klassen er klasse A. Når Taber-klassen har fundet en taber, aktivere den Vinder-klassen som derefter finder spillet vinder.

Information Expert
Information Expert-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. Vinder-klassen og Taber-klassen kan beskrives som information expert-klasser, da de indeholder alle de informationer de selv skal anvende.

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 en lettere forståelig kode som også nemt kan redigeres.
I programmet er Spil controlleren som starter programmet.

Polymorphism
Princippet polymorhism går ud på, at objekter kan ændre på deres form.

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.

landOnField
landOnField er en abstrakt metode som bliver oprettet i AbstractFelter-klassen. Metoden får tildelt en krop som indeholder kode, der giver det en opgave at udføre. Dette sker i underklasserne af AbstractFelter. Klasserne har hver deres landOnField-metode, som er specificeret til dem. landOnField-metoden bestemmer hvilke metoder klassen har, som f.eks. kunne være at feltet kan købes eller der skal betales husleje.[1]

[1] Genbrugt fra forrige opgave

4.2 Beskrivelse af klasser og objekter

Kast
I klassen Kast, er der udviklet en metode til at oprette en terning. Metoden returnerer også værdien af terningen efter at der er blevet slået med den.

Terning
Terning-klassen anvendes til at angive en værdi for en terning. Klassen er kodet således at den opretter én seks-siddet terning med værdi fra 1-6.

Spiller
Spiller er den klasse som indeholder metoderne til at styre spillerens placering og desuden om spilleren er i fængsel, har et kom gratis ud a fængsel kort, har fødselsdag, om spilleren er berettiget til en gratis ejendom og om spilleren har passeret start. Spiller klassen bruger også en metode til at bruge klassen kast til at slå med terningen. Spiller klassen indeholder også metoderne til at ændre i pengebeholdning, anlægs aktivitet værdi samt at tilføje eller fjerne et skøde.

Bræt
Denne klasse bruges til at danne de visuelle felter til Felt-klassen hvor spillerens brik kan lande på, alt efter værdien af terningkastets værdi. Bræt-klassen indeholder en tilhørende tekst og billede/farve for hvert felt på spillebrættet.

Vinder
Denne klasse har ansvaret for at finde vinderen af spillet. Dette gør den ved anvendelse af metoder som finder spilleren med den største likvideBeholdning. Denne klasse aktiveres ved hjælp af Taber-klassen, når en spiller har tabt.

AbstractFelter
AbstractFelter-klassen er en super-klasse til alle felt-typerne på brættet.

lanOnField landOnField-metoden er en abstrakt metode, som uddeler metoder til hver af de forskellige typer af felter på brættet.

4.3 Abstract

En klasse der indeholder én eller flere abstrakte metoder kaldes for en abstrakt klasse. En metode som endnu ikke er defineret kalder man for en abstrakt metode. De abstrakte metoder defineres i subklasser, som arver metoder fra den abstrakte klasser. Et eksempel kunne være Konto som en abstrakt klasse og opsparingskonto kunne være dennes subklasse. Konto kan ikke instantieres af sig selv, hvormed opsparingskonto kan instantieres ved at anvende metoder fra konto. I projektet er AbstractFelter en abstrakt klasse som indeholder den abstrakte metode landOnField.

4.4 Arv

I programmering betyder arv at der findes en "forælder"-klasse (også kaldet super klasse) og en "børne"-klasse. Børneklassen arver dermed indholdet, hvilket er metoderne, fra superklassen som den har mulighed for at udvide. Ved udvidning er der tale om at tilføje yderligere metoder end dem fra superklassen. Arv kan desuden beskrives som specialisering og generalisering. Specialisering går ud på f.eks. at en nedarvet klasse (børneklassen) er en bestemt type af superklassen. Her kunne superklassen hedde bygning, hvor en nedarvet klasse kunne hedde skole. Skole vil arve en del af de samme metoder fra bygning, men bygning vil indeholde flere metoder som specificere til skole. Dertil kan der tilføjes endnu en nedarvning, som kunne hedde privatskole eller kostskole. Dette er endnu en specialisering, hvormed man bliver mere detaljeret.
En generalisering vil sige at superklassen har metoder som tager hensyn til alle former for nedarvede

klasser. Dette gør at man fjerner de specifikke metoder og attributter fra privatskole, for at man kan danne generelle metoder der passer til alle former for skoler.
Alle felt typerne (Chance, Passiv, Fængsel og Ejendom) i programmet er nedarvede fra AbstractFelterklassen.

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 White Box Testing

5.1.1 Junit test-1

De udførte Junit test er inkulderet i programmet.

Test Case ID	TC01
Referat	Automatiseret test af addLikvidemidler
Krav	Beholdningen bliver opdateret med den givne mængde
Betingelser før	Likvidemidler er 100
Betingelser efter	Beholdningen 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	10
Forventet resultat	110
Faktisk resultat	110
Status	Bestået
Testet af	Christian Høj
Dato	29/11-2017
Testmiljø	Eclipse 4.7.0 på Windows 8

Tabel 14: Test case 1

5.1.2 Oversigt over inddata til test-1

Test data	Forventet resultat	Faktisk resultat	Status
-10	90	90	Bestået
10	110	110	Bestået
0	100	100	Bestået
abc	Fejl	Fejl	Bestået
-101	-1	-1	Bestået

Tabel 15: Inddata til test 1

5.1.3 Screenshot af test

```
/**Test af addLikvidemidler
 * addLikvideMidler(int likvideMidler)
 * Aktivbeholdning(100) - Likvidemidler er sat til 100 på forhånd
 * Positiv: tilføjer 10 til likvidemidler, expected 110, testen går igennem
 * Negativ: tilføjer 20 til likvidemidler, expected 110, testen går ikke igennem da actual=120
 */
@Test
public void testAddLikvidemidler() {
    beholdningTest.addLikvideMidler(10);
    int actual = beholdningTest.getLikvideMidler();
    int expected = 110;
    assertEquals(actual, expected);
}
```

Figur 6: Screenshot af addLikvideMidler metoden

5.1.4 Junit test-2

Test Case ID	TC02
Referat	Automatiseret test af harTabt
Krav	Spilleren skal erklæres som taber hvis spillerens beholdningsværdi kommer under 0
Betingelser før	Spillerens beholdningsværdi er 10
Betingelser efter	Spilleren er ikke erklæret for taber
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	false
Faktisk resultat	false
Status	Bestået
Testet af	Christian Høj
Dato	29/11-2017
Testmiljø	Eclipse 4.7.0 på Windows 8

Tabel 16: Test case 2

5.1.5 Oversigt over inddata til test-2

Test data	Forventet resultat	Faktisk resultat	Status
-10	false	false	Bestået
-11	true	true	Bestået
-9	false	false	Bestået

Tabel 17: Inddata til test 2

5.1.6 Screenshot af test

```
/**Test af addLikvidemidler
 * addLikvideMidler(int likvideMidler)
 * Aktivbeholdning(100) - Likvidemidler er sat til 100 på forhånd
 * Positiv: tilføjer 10 til likvidemidler, expected 110, testen går igennem
 * Negativ: tilføjer 20 til likvidemidler, expected 110, testen går ikke igennem da actual=120
 */
@Test
public void testAddLikvidemidler() {
    beholdningTest.addLikvideMidler(10);
    int actual = beholdningTest.getLikvideMidler();
    int expected = 110;
    assertEquals(actual, expected);
}
```

Figur 7: Screenshot af harTabt metoden

5.1.7 Junit test-3

Test Case ID	TC03
Referat	Automatiseret test af testHvemVinder
Krav	Den rette vinder findes
Betingelser før	Likvide midler er hhv. 20 og 21
Betingelser efter	Vinderen er fundet
Testens fremgangsmåde	1. Der oprettes en Junit test case. 2. Der opstilles en passende testmetode. 3. Junit test casen køres i Eclipse.
Test data	Spiller 1: 20 og Spiller 2: 21
Forventet resultat	Spiller 2 vinder
Faktisk resultat	Spiller 2 vinder
Status	Bestået
Testet af	Gunn Mohr Hentze
Dato	01/12-2017
Testmiljø	Eclipse 4.6.2 på Windows 10

Tabel 18: Test case 3

5.1.8 Screenshot af test

```
/**Test af testHvemVinder metoden.
 * testHvemVinder(Array af spillere)
 * Positiv: Likvidemidler sættes til hhv. 20 og 21 for spiller 1 og spiller 2. Testen godtages, da spiller 2 vinder
 * Negativ: Likvidemidler sættes til hhv. 20 og 21 for spiller 1 og spiller 2. Testen fejler, da spiller 1 vinder
 */
@Test
public void testTestHvemVinder() {
    Spiller spiller1 = new Spiller("Spiller 1") ;
    Spiller spiller2 = new Spiller("Spiller 2");
    spiller1.ændrLikvideMidler(20);
    spiller2.ændrLikvideMidler(21);
    String actual = vinderTest.testHvemVinder(new Spiller[] {spiller1, spiller2});
    String expected = spiller2.getNavn()+" har vundet";
    assertEquals(actual, expected);
}
```

Figur 8: Screenshot af harTabt metoden

5.2 Drift test

5.2.1 Databar test

Følgende fremgangsmåde er udført på DTUs databar, for at bekræfte om programmet faktisk virker på det system det skal kører på:

1. Installer programmet til PCen eller hav det på USB
2. Eksekver programmet filen 37_DEL3.JAR
3. Spillet startes op og brugerene skal indtaste spillerenes navne
4. Spillet er nu klar til at blive spillet

5.3 Black Box Testing

5.3.1 Brugertest

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?	

Tabel 19: Spørgeskema til brugertest

Brugertesten er udført af personer uden kendskab til projektet eller indmaden af projektet. Testen er udført af brugerene uden interaktion fra projektets deltagere, dvs. test-brugerene 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

Spørgsmål	Svar
Forstår du hvad spillet går ud på, uden producenternes hjælp?	Ja, men kun fordi jeg havde kendskab til matador-spillet
Har du brug for en brugervejledning for at spille spillet?	Nej, ikke rigtig, jeg forstår hvad der sker.
Er spillets tekster for at gennemføre spillet forståelige?	Ja
Er spillet brugervenligt?	Ja
Tog det for lang tid at spille spillet færdigt?	Det tog okay lang tid, men det er jo matador
Har du evt. forbedringer til spillet?	I kunne indsætte en vejledning til spillet, som også fortalte formålet med spillet.

Tabel 20: Brugertest 1

Yderligere bemærkninger fra testpersoner:

- Spillet kan blive kedeligt i længden, men sjovt til lidt småhygge.

- Kan godt lide at der kommer ens bils farve på de felter man har købt.
- Der kunne godt have været en brugervejledning i starten af spillet eller instrukser.

6 Brugervejledning

Programmet eksekveres ved at køre 37_DEL3.JAR. Dernæst indtastes spillerenes navne og spillet er nu klar til at begynde.

6.1 Minimumskrav

Følgende skal minimum være til rådighed for at køre programmet:
To brugere, mus, tastatur, skærm, 5 MB ledigt plads på harddisken, Java 7 eller nyere, Eclipse Standard Version: (4.5.0)

6.2 Import af programmet i Eclipse (Oxygen)

1. Åbn Eclipse
2. Tryk på "file"og vælg "import"
3. Under "General"vælg "Existing Projects into Workspace"og tryk "Next"
4. Tjek om der er et hak i "Select archive file"
5. Find filen på computeren og vælg den
6. Tryk på "Finish"

6.3 Import fra Github

1. Åbn Eclipse
2. Tryk på "File"og vælg dernæst "Import"
3. Under "Git"vælges "Projects from git"og herefter på "Next"
4. Vælg "Clone URL"og tryk så "Next"
5. Indsæt https://github.com/Christianhoej/CDIO3 i feltet "URL"og tryk derefter på "Next"
6. Tjek om der er et hak i "master"
7. Vælg hvor du vil have projektet
8. Tjek at der er hak i "Import existing Eclpise projects"
9. Tryk på "finish"

6.4 Eksekver programmet i Eclipse

1. Åbn Eclipse (programmet er installeret og importeret til Eclipse)
2. Vælg projektet i Package Explorer (venstre side i Eclipse standard view)
3. Vælg controller pakken under projektet "37_DEL3"
4. Højreklik derefter på SpilSpil Klassen og vælg "run as..."og vælg "1 Java Application"

7 Konklusion

7.1 Proces

IOOuterActive har haft en god arbejdsprocess med at anlysere, designe, implementere, teste og dokumentere. Processen har været flydende og har overholdt de fleste af internt satte deadlines.

7.2 Produkt

IOOuterACtive har lavet et program som simulerer et Monopoly Junior spil. Produktet er blevet rigtigt godt og opfylder alle de krav som der blev stillet til at sarte med samt supplerende krav som der i samarbejde med DTU er blevet stillet.

7.3 Perspektivering

Sammenlignet med et rigtigt Monopoly Junior spil kan det spil IOOuterActive har udviklet virke lidt kedeligt da alt mann skal gøre som spiller er at trykke på en knap og så fortager systemet det hele. Dette gø at man ikke får samme "hands-on"følelse som man ville få hvis man sad med pengene og terningen i hånden i stedet.

7.4 Samlet

Samlet mener vi i IOOuterActive at vi har lavet et godt produkt med tilfredstillende dokumentation og arbejdsprocess der tilfredsstiller alle kundens krav.

8 Litteraturliste

1. Lewis Loftus: Java Software Solutions, Foundations of Program Design. 7. udg. Pearson, 2012. (Bog)
2. Larman, Craig: Applying UML and Patterns. 1. udg. Addison Wesley Professional, 2004. (Bog)
3. Creating a Repository: Git Eclipse. DrBFraser. Fraser, Brendan . 21.04.2013. Internetadresse: https://www.youtube.com/watch?v=r5C6yXNaSGo - Besøgt d. 01.11.2017 (YouTube)
4. Inge-Lise Salomon: Undervisning. DTU, 04.09.2017
5. Mads Nyborg: Undervisning. DTU, 05.09.2017
6. Daniel Kolditz Rubin-Grøn: Undervisning. 08.09.2017

9 Bilag

9.1 Bilag 1

Projekt Navn								
Time-regnskab	**Ver. 2017-12-01**							
Dato	**Deltager**	**Analyse**	**Design**	**Impl.**	**Test**	**Dok.**	**Andet**	**Ialt**
13-11-2017	Gunn		0,5				2	2,5
-	Christian						2	2
-	Adam						1	1
-	Janus						1	1
-	Ahad						1	1
14-11-2017	Ahad	2						2
-	Gunn	2						2
-	Christoffer	2						2
-	Janus	2						2
-	Adam	2						2
16-11-2017	Janus	3						3
-	Adam	3						3
-	Gunn	3						3
18-11-2017	Adam	3						3
-	Gunn	3						3
-	Christian						1	1
19-11-2017	Janus		3					3
20-11-2017	Christian	1,5						1,5
-	Gunn	1						1
-	Ahad	1						1
21-11-2017	Christoffer	3						3
-	Christian		3					3
-	Adam		3		1			4
-	Janus		3					3
-	Gunn	3						3
-	Ahad	3						3
22-11-2017	Christian			1,5				1,5
-	Adam			1,5				1,5
23-11-2017	Christian			2				2
-	Adam			2				2
-	Janus					1		1
-	Gunn			2				2
24-11-2017	Janus			1,5				1,5
-	Ahad			2				2
25-11-2017	Janus				2			2
26-11-2017	Christian		2					2
-	Adam					2		2
27-11-2017	Christian		1		1	1		3
-	Christoffer		3					3
-	Janus		1					1
-	Adam		3	4				7
-	Gunn			7				7
-	Ahad		3					3
28-11-2017	Christoffer		3					3
-	Christian		2			1		3
-	Adam					2		2
-	Janus			3				3
-	Gunn			3				3
-	Ahad					3		3
29-11-2017	Christoffer		2					2
-	Christian				3,5			3,5
-	Adam			3,5				3,5
-	Ahad					3,5		3,5
-	Gunn			3,5				3,5
30-11-2017	Gunn			8		2		10
-	Ahad			3	3	2		8
-	Janus			8				8
-	Christoffer		3				3	6
-	Christian			3	3	2		8
01-12-2017	Ahad				2			2
-	Gunn			8	1			9
-	Janus			3				3
-	Christoffer				2	1		3
	Sum		**35,5**	**69,5**	**18,5**	**20,5**	**11**	**132,5**