CDIO3

Gruppe 16

10. november 2017



Mathias Fager s175182



Milishia Moradi s175193



Nicki Christiansen s170208



Semi Seitovski s175181



Simon Pedersen s175195



Thyge S. Steffensen s175176

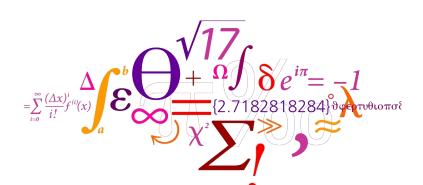


Timeregnskab

Timeregnskabet kan ses via linket <u>her</u> eller i billag XX.

Deltager	Total timer per deltager
Mathias Fager	3
Milishia Moradi	3,5
Nicki Christiansen	10
Semi Seitovski	6
Simon Steen Pedersen	5,5
Thyge Steffensen	14,5
Total	42,5

Figur 1: Overblik over Timeregnskab den 21/11/2017



Indhold

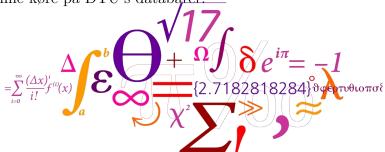
1	Ana	alyse
	1.1	Kravliste
	1.2	UseCases
	1.3	UseCase diagram
	1.4	GRASP
2	Des	$_{ m sign}$
	2.1	Klasse diagram
	2.2	Sekvensdiagram
	2.3	System sekvensdiagram
	2.4	Domænemodel
3	Dol	kumentation 11
	3.1	Forklar hvad arv er
	3.2	Forklar hvad abtract betyder
	3.3	Fortæl hvad det hedder hvis alle fieldklasserne har en landOn-
		Field metode der gør noget forskelligt
	3.4	Dokumentation for test med screenshots
	3.5	JUnit test
	3.6	Positiv negativ test
	3.7	Black- og Whitebox test
	3.8	Dokumentation for overholdt GRASP
	3.9	Vejledning til import af Git Repository til eclipse



1 Analyse

1.1 Kravliste

- 1. Spillet skal være mellem 2 4 personer
- 2. Spillerne skal slå terninger på skift
- 3. Der skal udskrives en tekst der omhandler det aktuelle felt, når spilleren lander på et felt
- 4. Spilleren skal købe feltet, hvis der landes på et frit felt.
- 5. Spilleren skal betale husleje til ejer, hvis der landes på et købt felt
- 6. Spilleren skal betale dobbelthusleje til ejer, hvis der landes på et "par".
- 7. Spillere modtager 2M, når start passeres.
- 8. spilleren ryger direkte i fængsel, hvis der landes på "GÅ I FÆNGSEL". Der modtages ikke 2M, når start passeres.
- 9. Sidder man i fængsel, bliver ens "Du løslades uden omkostninger-kort brugt. Hvis kortet ikke haves betales der 1M.
- 10. Landes der på "Chance", bliver effekten printet og den printede effekt sker.
- 11. Landes der på "Gratis parkering"sker ingenting, og turen går videre.
- 12. Landes der på "På besøg"sker ingenting, og turen går videre.
- 13. Spillerne starter med en balance på:
 - (a) 16, hvis der er 4 spillere.
 - (b) 18, hvis der er 3 spillere.
 - (c) 20, hvis der er 2 spillere.
- 14. Spillet slutter når den første spiller har mistet alle sine penge.
- 15. Spilleren med flest penge vinder, når spillet slutter, og der printes "Tillykke "vinder", du har vundet!"
- 16. Spillerne skal kunne gå flere omgange rundt på spillepladen.
- 17. Spillet skal kunne køre på DTU's databarer.



1.2 UseCases

UseCase Section: Start af spil	Comment
Scope	Monopoly Junior spil af IOOuterActive
Level	User-goal
Primær Aktør	IOOuterActive
Stakeholder og interes-	IOOuterActive er interesseret i at spillerne skal kunne
senter	starte spillet.
Forudsætninger	Spillet er installeret på enheden.
Success guaranti	Spillet starter og spillerne bliver sendt videre til opsæt-
	ning.

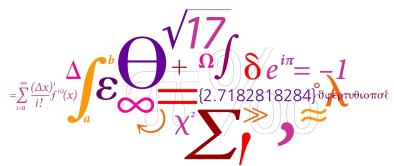
UseCase Section: Op- sætning af spil	Comment
Scope	Monopoly Junior spil af IOOuterActive
Level	User-goal
Primær Aktør	Spillerne
Stakeholder og interes-	Spillerne er interesserede i at kunne vælge antal spillere
senter	og deres brikker.
Forudsætninger	Spillet er startet op og spillerne har nu mulighed for at vælge antal spillere og ønskede brikker.
	0 1
Success guaranti	Der er blevet valgt antallet af spillere og hver spiller har valgt sin brik, herefter er spillet klar til at blive spillet.



Fully dressed UseCase:

UseCase Section: Spil-	
lerne slår med ternin-	Comment
gerne	
Scope	Monopoly Junior spil af IOOuterActive
Level	User-goal
Primær Aktør	Spillerne
Stakeholder og interes-	Spillerne er interesseret i at kunne trykke på en knap,
senter	og få et billede af en terning med tilfældige værdier.
Forudsætninger	Spillet er startet op og spillerne har valgt antallet spillere
Fordusætninger	og deres ønskede brikker.
Success guaranti	Der er blevet valgt antallet af spillere og hver spiller har
Success guaranti	valgt sit navn, herefter er spillet klar til at blive spillet
Hoved succes scenarie	Spillerne får vist terningens øjne og deres brik bliver
	rykket hen til det respektive felt.
Alternative udfald	Negative udfald:
	- IOOuterActive har opdateret spillet og derved opstår
	der en fejl når spillerne slår med terningen, der kan ende
	i at der ikke slået med ternignen.
	- Systemet blokerer for en spillers tur
	- En spiller hopper fra/på og derved skal spillet startes
	om.
Specielle krav:	- Enheden som spillet kører på skal kunne køre Java.
	- Spillerne skal kunne interagere med GUI'en ved brug
	af mus eller touch.
	- Der skal være plads på enheden til at kunne hente
	spillet.
Hyppighed	Hver tur bliver der slået med en terning.

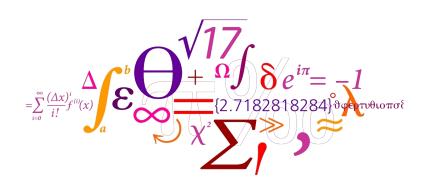
UseCase Section: En spiller køber et felt	Comment
Scope	Monopoly Junior spil af IOOuterActive
Level	User-goal
Primær Aktør	Spillerne
Stakeholder og interes- senter	Spillerne er interesseret i at købe det aktuelle felt.
Forudsætninger	Spillet er i gang og en spiller har slået med terningerne
Success guaranti	Spilleren køber og ejer nu feltet



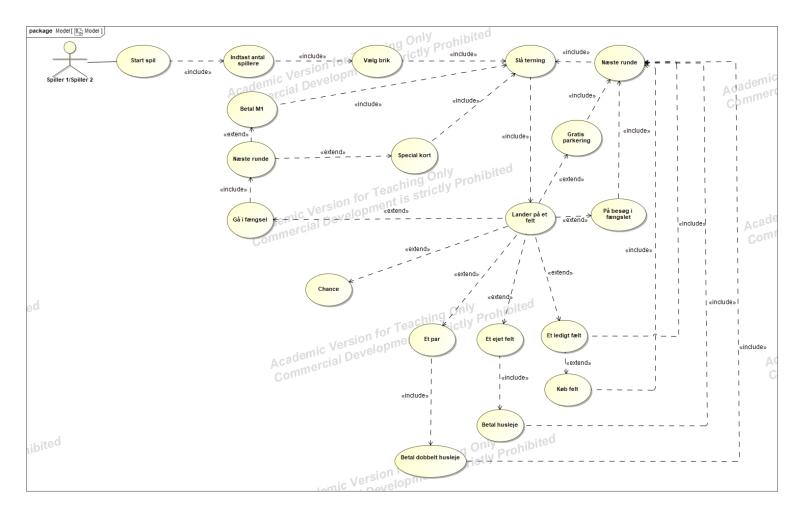
UseCase Section: Spiller lander på et ejet felt	Comment
Scope	Monopoly Junior spil af IOOuterActive
Level	User-goal
Primær Aktør	Spillerne
Stakeholder og interes-	****
senter	
Forudsætninger	Spillet er i gang og en spiller lander på et felt, som er
rorudsætninger	ejet af en anden spiller.
Success guaranti	En spiller lander på et felt, der er ejet af en anden spiller
Success guaranti	og får derfor en negativ effekt på balancen.

UseCase Section: En spiller taber	Comment
Scope	Monopoly Junior spil af IOOuterActive
Level	User-goal
Primær Aktør	Terningerne
Stakeholder og interes-	Spillerne er interesseret i at deres balance ikke når 0 og
senter	derved taber.
Forudsætninger	Spillerne har slået med terningen.
Success guaranti	En spilleres balance når 0, taber de spillet og spillet afsluttes.

UseCase Section: Spillet afsluttes	Comment
Scope	Monopoly spil af IOOuterActive
Level	User-goal
Primær Aktør	IOOuterActive
Stakeholder og interes-	IOOuterActive er interesserede i at programmet viser en
senter	vinder og afsluttes.
Forudsætninger	En spiller har nået en balance på 0 eller derunder.
Success guaranti	Spillet viser en vinder og kan derefter afsluttes.



1.3 UseCase diagram

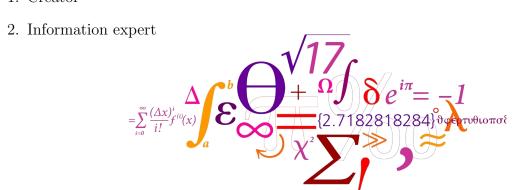


Figur 2: UseCase diagram tegnet i MagicDraw

GRASP 1.4

GRASP står for General Responsibility Assignment Software Patterns. GR-ASP bruges til at give det rigtige ansvar til de forskellige klasser, der bliver oprettet under udviklingen af et program. GRASP indeholder 9 patterns. Patterns bliver brugt til at strukturere et problem, samt at finde en passende løsning. De 9 patterns er:

1. Creator



- 3. Low coupling
- 4. Controller
- 5. High cohesion
- 6. Indirection
- 7. Polymorphism
- 8. Protected variations
- 9. Pure fabrication

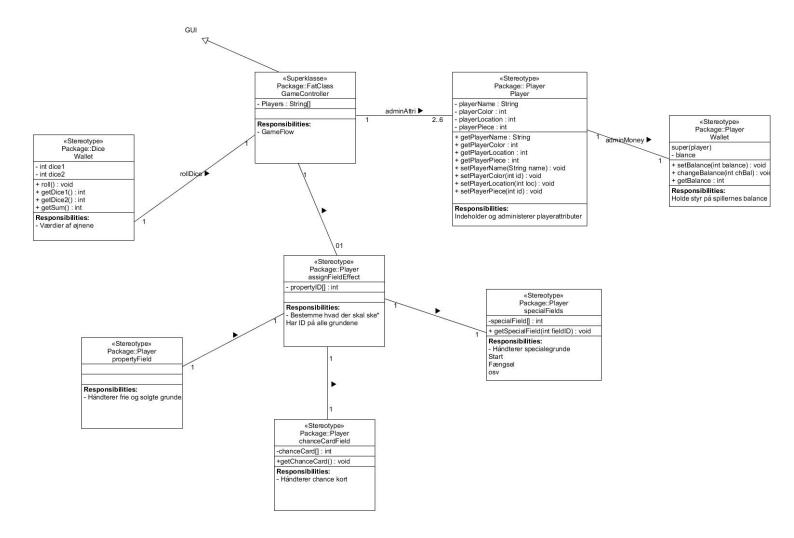
(Der skrives mere når vi er nået længere i projektet)



2 Design

Herunder ses en række 'design steps', som skal hjælpe os med at lave Monopoly Junior spillet.

2.1 Klasse diagram



Figur 3: Klasse diagram tegnet i UMLet

Klasse diagrammet bygger på vores umiddelbare overvejelser, såvel som vores use case's. Dette er for at illustere sammenspillet mellem vores klasser og deres associationer. Dog skal det siges, at dette er en skitse og den aktuelle programmering kan variere heraf.

2.2 Sekvensdiagram

Vi har her lavet et sekvensdiagram, der skal skabe et overblik over hvordan aktøren, her spilleren, kommunikerer med spillet.

2.3 System sekvensdiagram

Vi har her lavet et systemsekvensdiagram, for at forhøje gennemsigtigheden ved bruge af 'chanceCard' klassen.

2.4 Domænemodel

Ved hjælp af domænemodellen vil vi trække paraleller mellem den virkelige verden og programmeringen. Domænemodellen er en visuel repræsentation af konceptklasser og 'objekter fra den virkelige verden'. Ved hjælp af denne kan vi også oplyse kunden om, hvad vi vil lave.



3 Dokumentation

3.1 Forklar hvad arv er

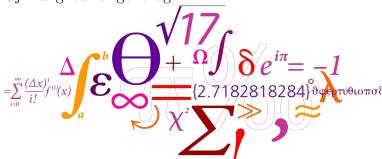
Et af de vigtigste elementer i objektorienteret programmering er nedarvning, men hvad betyder nedarvning? At arve betyder i bund og grund at noget med en materiel værdi overdrages til noget eller nogle efterkommere. I programmerings verden ville man hellere definere arv som en eller flere egenskaber der overføres fra en generation til en anden generation, også sagt på en anden måde, at en klasse der får overført egenskaber fra en anden klasse. Den klasse der nedarves fra kaldes for en Superklasse (parent), hvor klassen der nedarver eller får egenskaber fra en Superklasse, kaldes for en Subklasse (child). (Kressner, Marts 2014) Ved benyttelse af denne proces vil oplysningerne om de forskellige klasser ende op i en hierarkisk rækkefølge. På figuren ses forskellige eksempler på typer af nedarvning der findes, men dog skal man være opmærksom på, at Java ikke understøtter flere nedarvninger. I Java kan en klasse kun have én Superklasse, dvs. at hver klasse kun kan nedarve fra én klasse. (Kressner, Marts 2014)

3.2 Forklar hvad abtract betyder

Den abstrakte klasse kører ingen metoder og indeholder kun attributter. Klassen bruges som en slags "over"-klasse, hvor andre klasser nedarver attributterne. Det anvendes i situation, hvor man f.eks har to klasser, der har mange af de samme attributter. Så ville alle de attributter som de har filfælles stå i den abstrakte klasse. Hvis man f.eks har klasserne lærer og elev, kunne den abstrakte klasse til disse, hedde "personer", da både lærer og elever har en fødselsdato, køn, navn osv.

3.3 Fortæl hvad det hedder hvis alle fieldklasserne har en landOnField metode der gør noget forskelligt

Hvis alle fieldklasserne gør brug af den samme land On
Field metode, er det fordi denne metode er en super metode. En super metode er nedarvet til de forskellige subklasser fra super klassen, dette gør at alle klasserne kan gøre brug af samme metode. Eksempeltvis, hvis vi har en masse dyr som klasser, kat, hund, kanin etc. så kan de alle nedarve super metoden eat() fra superklassen som hedder Survival
Requirements, eftersom disse er metoder alle dyrene får brug for, så vil det give mening at lave det til en superklasse med supermetoder , så der holdes lav kobling og høj kohæsion, samt undgås kopiring af kode og høj mulighed for genbrug.



3.4 Dokumentation for test med screenshots

3.5 JUnit test

JUnit test er en autonomisreret testmetode. Her skriver/koder man selv en test, som tester java kode. Oftest opbygger man JUnit test ud fra Java klasser. Der er mange måder hvorpå man kan bruge JUnit testen. Man kan både skrive testen inden, man kan skrive den efter, man kan lave den på baggrund af indsigt i koden eller uden nogen form til kendskab af programkoden. Det to sidst nævnte kaldes Black- og Whitebox test. Her er et eksempel på et stykke udført JUnit test fra vores spil: Det ser på Figur 3.5 at koden ikke bestod testen, hvor vi også får vist den rette, selv skrevet, fejl besked.

3.6 Positiv negativ test

Denne test bruges for at teste om systemet kan håndterer 'ugyldige' input, dette kan både være negative som positive værdier, såvel som bogstaver og andre tegn. Denne test forbindes oftest med en eller flere ækvivalensklasse test.

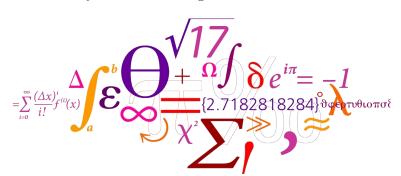
3.7 Black- og Whitebox test

Blackbox test: Her har du intet kendskab til koden, ud over hvad denne skal kunne gøre. Det er derfor –lige til højre benet– at teste det faktiske output, mod det forventede output og at teste ækvivalensklasserne. Whitebox test: Her opstiller man tests på baggrund af koden og dens opbygning. Man vil med disse tests teste hele koden, altså alle instruktioner, forgreninger og stier, som koden kan blive udført i. Dette er ikke altid muligt med BlackBox testen, da man her ikke har et indblik i koden.

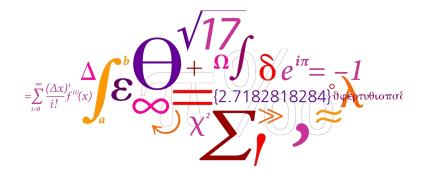
3.8 Dokumentation for overholdt GRASP

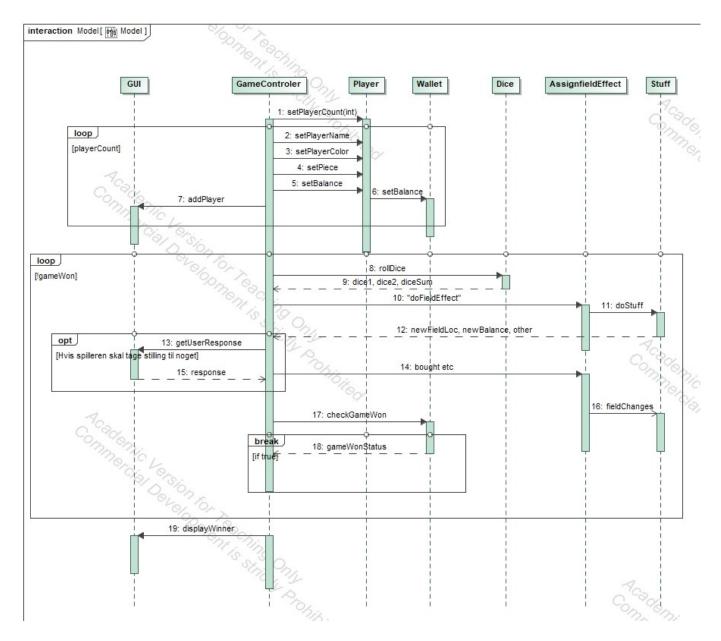
3.9 Vejledning til import af Git Repository til eclipse

Man skal i første omgang have et URL af det pågældende repository, som du kopierer. Derefter åbner du eclipse og åbner menuen Window \Rightarrow View \Rightarrow søg på "git" \Rightarrow git repositories. Derefter har du vinduet "Git Repositories", hvor der er en knap; "Clone a git repository and add to this view". Ikonet er mappen i midten med en blå pil på, her trykker du. Hvis du huskede at kopiere URL'et trykker du nu "Next", hvorefter at alle branches bliver loaded. Når den har loaded trykker du next igen. Nu kommer du til det



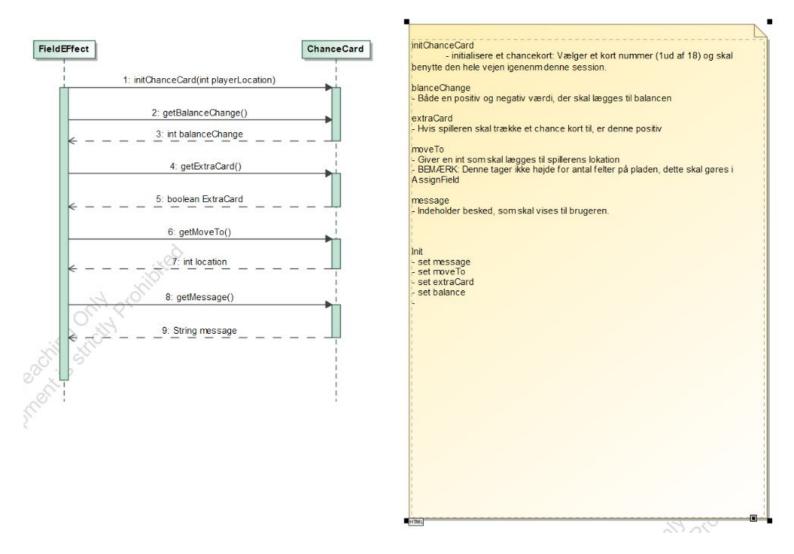
sidste vindue, hvor du trykker "Finish, og nu har du importeret(clonet) git repositoriet.





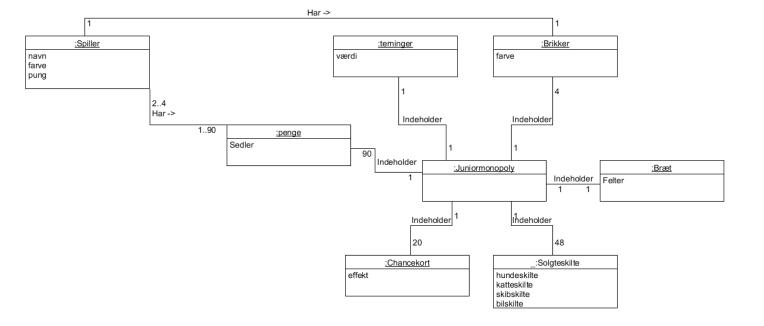
Figur 4: Sekvensdiagram tegnet i MagicDraw



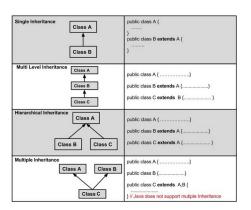


Figur 5: Systemsekvensdiagram tegnet i MagicDraw





Figur 6: Domænemodel tegnet i UMLet



Figur 7: eksempel på typer af arv



fig/JUnitTestDice.jpg

Figur 8: J Unit test udført i Eclips
e ${\rm den}\ 23/11\text{-}2017$

