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

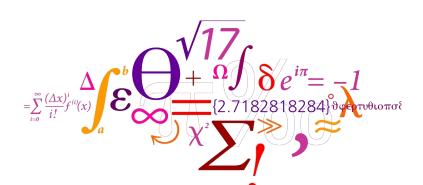


Timeregnskab

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

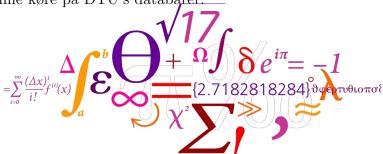
1	Ana	lyse				
	1.1	Kravliste				
	1.2	UseCases				
	1.3	UseCase diagram				
2	Desi	ign 8				
	2.1	Klasse diagram				
	2.2	Sekvensdiagram				
	2.3	System sekvensdiagram				
	2.4	Domænemodel				
3	Dokumentation 1					
	3.1	Forklar hvad arv er				
	3.2	Forklar hvad abtract betyder				
	3.3	Fortæl hvad 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 15				
	3.10	Brugertests				



1 Analyse

1.1 Kravliste

- 1. Spillet skal være mellem 2 4 personer.
- 2. Spillerne skal slå med en terningen 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 ejeren, hvis der landes på et købt felt.
- 6. Spilleren skal betale dobbelthusleje til ejeren, hvis der landes på en "par-ejet hustand.
- 7. Spillere modtager 2M, når start passeres.
- 8. Spilleren ryger direkte i fængsel, hvis der landes på "GÅ I FÆNGSEL". Der modtages ikke 2M, hvis start passeres i forbindelse med dette.
- 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 vinder. Efterfølgende printes "Tillykke (spiller)", 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: Opsætning af spil	Comment	
Scope	Monopoly spil af IOOuterActive	
Level	User-goal	
Primær Aktør	Spillerne	
Stakeholder og interes-	Spillerne er interesserede i at kunne starte spillet ved at	
senter	vælge antal spillere og deres brikker	
Forudsætninger	Spillet bliver kørt, og spillerne har nu mulighed for at vælge antal spillere og ønskede brikker	
Success guaranti	Der er blevet valgt antallet af spillere, og hver spiller har	
Duccess guaranti	valgt sin brik, herefter er spillet klar til at blive spillet	



Fully dressed UseCase:

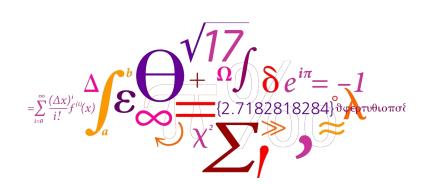
II. C. C. C. C. C.			
UseCase Section: Spil-			
lerne slår med ternin-	Comment		
gerne			
Scope	Monopoly spil af IOOuterActive		
Level	User-goal		
Primær Aktør	Spillerne		
Stakeholder og interes-	Spillerne er interesseret i at kunne trykke på en knap,		
senter	og få et billede af to terninger med tilfældige værdier		
Form denotes in most	Spillet er startet op, og spillerne har valgt antallet spil-		
Forudsætninger	lere og deres ønskede brikker		
C	Der er blevet valgt antallet af spillere, og hver spiller har		
Success guaranti	valgt sit navn, herefter er spillet klar til at blive spillet		
TT 1 .	Spillerne får udgivet en værdi af to terninger, og lander		
Hoved succes scenarie	derefter på et felt		
Alternative udfald	Negative udfald:		
	- IOOuterActive har opdateret spillet, og derved opstår		
	der en fejl når spillerne slå med terningerne, der kan		
	ende i at der ikke bliver slået to terninger		
	- Systemet blokerer for en spillers tur		
	- En spiller hopper fra/på, og derved skal spillet startes		
	om		
G . 11 1	- Enheden som spillet kører på skal være kompatibel		
Specielle krav	med 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 terninger		



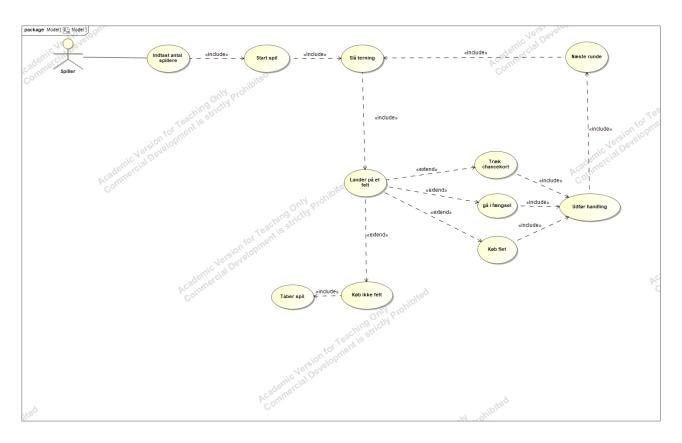
UseCase Section: Spiller lander på et felt	Comment	
Scope	Monopoly spil af IOOuterActive	
Level	User-goal	
Primær Aktør	Spillerne	
Stakeholder og interes-	Spillerne er interesseret hvilken effekt de kommer i møde	
senter	når de lander på et felt	
	- Hvis feltet ikke er opkøbt af nogen, kan spilleren der er	
	landet på feltet købe det, medmindre det er et specielt	
	felt	
	- Hvis feltet er ejet af nogen, skal spilleren der er landet	
	på feltet betale den der ejer feltet	
	- Feltet de lander på kan også være et fængsel, der gør	
	at spilleren mister en tur	
Forudsætninger	Spillet er i gang og en spiller har slået med terningerne	
Success guaranti	Spilleren køber og ejer nu feltet	

UseCase Section: Spil-			
ler trækker et chance-	Comment		
kort			
Scope	Monopoly spil af IOOuterActive		
Level	User-goal		
Primær Aktør	Spillerne		
Stakeholder og interes-	Spillerne er interesseret i hvilken bonus de får når de		
senter	trækker et chancekort		
Forudsætninger	r Spillet er i gang og en spiller trækker et chancekort		
Success guaranti	En spiller lander trækker et chancekort og får en		
Success guaranti	belønning eller en straf		

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	Alle spillere undtagen en, har fået en balance på 0	
Success guaranti Spillet viser en vinder og kan derefter afsluttes		



1.3 UseCase diagram



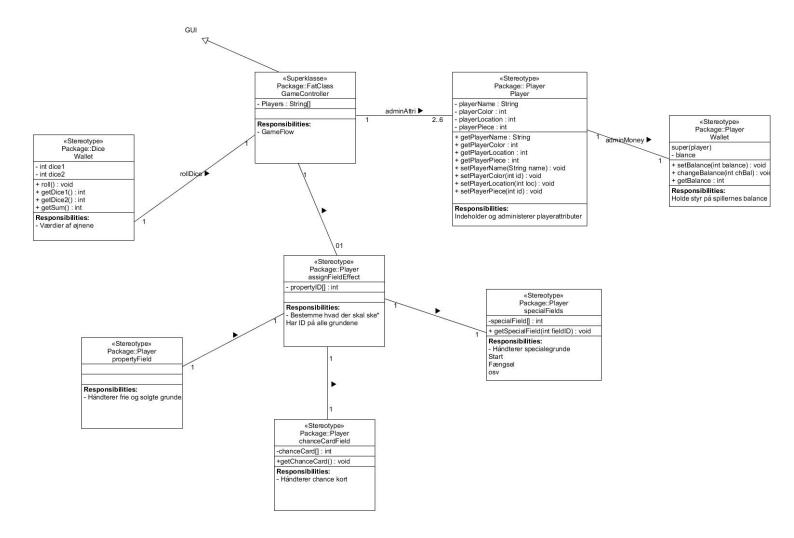
Figur 2: UseCase diagram tegnet i MagicDraw



2 Design

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

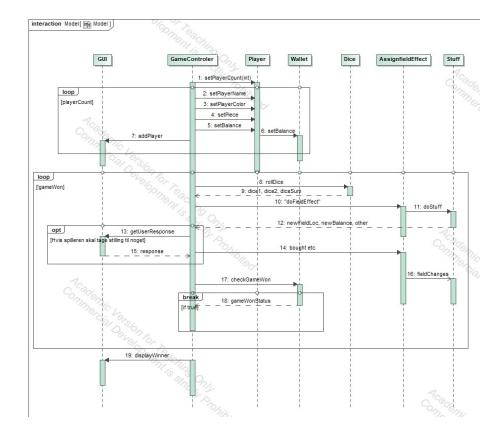
2.1 Klasse diagram



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

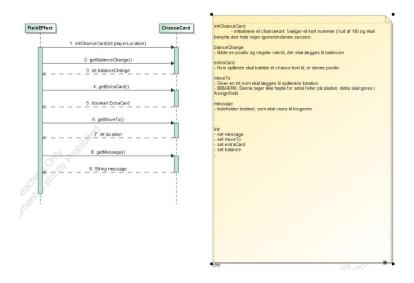


Figur 4: Sekvensdiagram tegnet i MagicDraw

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



2.3 System sekvensdiagram

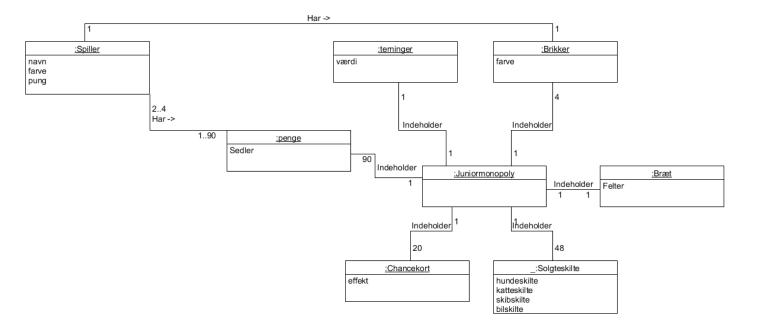


Figur 5: Systemsekvensdiagram tegnet i MagicDraw

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

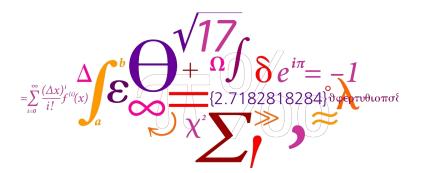


2.4 Domænemodel



Figur 6: Domænemodel tegnet i UMLet

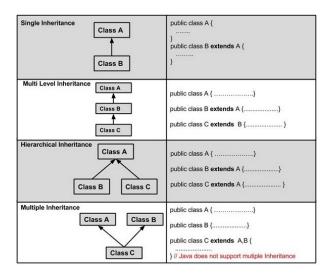
Ved hjælp af domænemodellen vil vi trække paraleller mellem den virkelige verden og programmeringen. Domænemodellen er en visuel repræsentation af konceptklasser og 'objekter fra den virkelige verden'. Ved hjælp af denne kan vi også oplyse kunden om, hvad vi vil lave.



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). Ved benyttelse af denne proces vil oplysningerne om de forskellige klasser ende op i en hierarkisk rækkefølge. På figur 7 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. ¹



Figur 7: eksempel på typer af arv

2

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 attri-

²https://www.tutorialspoint.com/java/java/nheritance.htm



 $^{^{\}rm 1}$ Java Programmering - En bog for begyndere af Henrik Kressner

butterne. Det anvendes i en situation, hvor man f.eks har to klasser, der har mange af de samme attributter. Så vil alle de attributter, som de har tilfælles 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 landOnField 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 SurvivalRequirements, 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

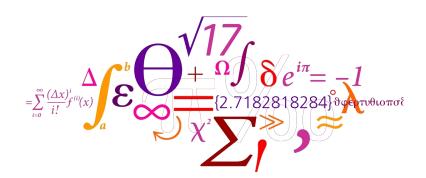
3.5 JUnit test

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 ?? 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.



```
| Tribled after 2001 seconds | Trible |
```

Figur 8: JUnit test udført i Eclipse den 23/11-2017

3.7 Black- og Whitebox test

Blackbox test: Her har man intet kendskab til koden, ud over hvad denne skal kunne gøre. Det er derfor lige til højrebenet, 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

GRASP står for General Responsibility Assignment Software Patterns. GRASP 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
- 2. Information expert
- 3. Low coupling
- 4. Controller
- 5. High cohesion
- 6. Indirection
- 7. Polymorphism
- 8. Protected variations
- 9. Pure fabrication

Disse patterns kan bruges som guidelines til at udvikle et software projekt. Her løser man problemer som ofte opstår under software udvikling, ved at gøre det overskueligt hvilken class der har ansvar for hvad. Vores Creator class er "GameController"som sørger for at "Player"class'ens information går videre til selve spillet. Information Expert er vores "Player"class, da den indeholder antallet af spillere og deres brikker.

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 ⇒ View ⇒ søg på "git" ⇒ 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.



3.10 Brugertests

Test person	Hvad man vil teste	Trin	Forventet resultat	Testresultat
Spillere	Antallet af spillere	Vælg an- tallet af spillere	Man kan være mellem 2 - 4 spillere	Pass
Spillere	Går turen videre	Spillerne slår med terninger- ne	En spiller slår med terningerne og turen går videre efter	Pass
Spillere	Udskrives der tekst når en spil- ler lander på et felt	Når en spiller lander på et felt	Der bliver printet en tekst der fortæller om feltet	Pass
Spillere	Køber spilleren feltet når han/hun lander på det	Når spillet er i gang	Feltet bliver opkøbt	Pass
Spillere	Spilleren er tvunget til at beta- le ejeren af feltet han lander på	Når spillet er i gang	Spilleren betaler ham/hen- de der ejer feltet	Pass



	D° :11 03.4			
Spillere	Får spilleren 2M når de passere start feltet	Når spillet er i gang	Spillerens balance stiger med 2M	Pass
Spillere	Ryger spilleren i fængsel og mi- stes der 2M	Når en spiller lander på "FÆNGSEL	Spilleren modtager ikke 2M	Pass
Spillere	Skal spilleren give sit omkost- ningskort eller betale 1M når han/hun er i fængsel	Når en spiller er på "FÆNGSEL	Spillerens mister sit omkost- ningskort eller betaler 1M "hvis kortet er brugt feltet	Fail
Spillere	Virker chance- kort	Når en spiller trækker et chancekort	Spilleren trækker et chance- kort og får en effekt	Fail
Spillere	Virker besøg feltet	Når en spiller lander på "Besøg" felte	Der sker intet og turen går videre t	Pass
Spillere	Kan spillerne gå rundt om pladen	Når spillerne har været en runde rundt på pladen	Spillerne kan gå flere om- gange rundt på pladen	Pass



