



Danmarks  
Tekniske  
Universitet

## CDIO Del 3

Afleveringsfrist: Fredag d. 25/11-2022

02312 / 62531 / 62532

Indledende Programmering, Versionsstyring og Testmetoder og  
Udviklingsmetoder til IT Systemer

Gruppe nr 1



s047560

Torben Storm Rasmussen

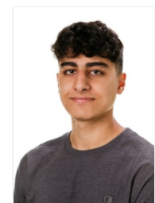
s221136

Ilias Elouali



Uffe Clausen  
s224275

Nabil Abbasi  
s224320



# 1.Timeregnskab

	T for Til stede, F for Fraværende .													
Gruppe 1 CDIO 1	14/11 man.	15/11 tir.	16/11 ons.	17/11 tor.	18/11 fre.	19/11 lør.	20/11 søn.	21/11 man.	22/11 tir.	23/11 ons.	24/11 tor.	25/11 fre.	Fraværende F	Til stede T
Nabil	T	T	T	T	T	T		T	T	T	T	T	0	10
Uffe	T	T	T	T	T	T		T	T	T	T	T	0	10
Torben	T	T	T	T	T	T		T	T	T	T	T	0	10
Ilias	T	T	T	T	T	T		T	T	T	T	T	0	10
Tid brugt på opgave:													Timer:	
Nabil	3	3	2	2	1.5	2		2	2	2	1.5	1.5	22.5	
Uffe	4	4	3	3	4	4		3	3	3	3	1.5	36.5	
Torben	4	4	3	3	4	4		3	3	3	3	1.5	36.5	
Ilias	3	3	2	2	1.5	2		2	2	2	1.5	1.5	22.5	

Figur 4.9 Billede af timeregnskab i Excel

## 2.Indholdsfortegnelse

1.Timeregnskab	2
2.Indholdsfortegnelse	3
3.Resumé	3
4.Indledning	4
5.Projektplanlægning	4
6.Krav/Analyse	6
7.Versionering	8
8.Design	11
9.Implementering	17
10.Test	18
11.Konklusion	26
12.Bilag	26

## 3.Resumé

Følgende rapport dykker ned i en udviklingsprocess af et spil monopoly junior. spilfirmaet IOOuterActive har fået til opgave at udvikle et spil, der kan spilles af 2-4 spillere og som der virker som et klassisk spil monopoly junior. For at løse denne opgave, har udviklerne valgt at planlægge dette projekt nøje på forhånd. Her har man lavet en risikoanalyse, for at vurdere sandsynligheden samt betydningen for adskillige risici og hvor man herudfra har slået fast, hvilke beslutninger man er villig til at tage og hvordan man forventer at anvende den tid man har fået givet. Derefter har man begået sig i en analyse, af de krav som man har fået stillet fra kundens side. Kravene har man betvivlet og fået specificeret i løbet af udviklingsprocessen under samtale med kunden. Man har derudover tolket på kundens visionsbeskrivelse, for bedst muligt at kunne sætte sig i kundens position mht. forventninger til det aspirerede produkt. Dernæst har man udnyttet Github med formålet om at få opsat nogle tilpassede rammer for udviklerne, så de i flydende samarbejde har haft mulighed for at arbejde på projektet på simultant vis. For så at gøre det tydeligt for både brugere og udviklerne selv, har man valgt at lave en konfigurationsstyring del. Denne del har til formål at tilgængeliggøre den info, der er nødvendig for brugeren, for at vedkommende kan tilgå forskellige dele af både produktet og koden til denne. For at visualisere produktets funktionalitet samt

opbygning, har man opstillet produktet i UML-diagrammer. Dette har man gjort både før og under udviklingsprocessen, for at klargøre hvordan forskellige dele af produktet er opbygget og hvordan disse fungerer. Derefter har man skrevet en implementerings del, for at vise hvordan at man har arbejdet i forskellige branches på IntelliJ og hvordan at man løbende har merget og pulled i Git-repository for at opretholde programmets funktionalitet og for at mindske sandsynligheden for at man får arbejdet på tværs af hinanden. For at sikre sig at det spil man har fået udviklet og dets funktioner virker optimalt og op til nogle bestemte grænser, har man opsat diverse test. Disse får man beskrevet i en test-del, der kommer næstsidst i rapporten. Til sidst har man med udgangspunkt i udviklernes refleksioner samt krav og analyse, udarbejdet en konklusion, for at få sat en ende på CDIO del 3.

*se i øvrigt afsnit 11. Konklusion side 27.*

#### **4.Indledning**

I samarbejde med IOOuterActive er en række motiver og spil blevet udviklet, der alle fokuserer på kundens behov og bedrifter. Efter at have udviklet en terningsfase og brætspils fase, har IOOuterActive endelig fået udleveret en opgave som samler samtlige faser til et. IOOuterActive har fået en opgave, hvor spillet Monopoly Junior skal udvikles. Opgaven er blandt andet blevet specificeret med forskellige krav, som tilsammen skal skabe de perfekte rammer for kundes vision af et fejlfrit Monopoly Junior spil. I denne opgave har vi fået analyseret, designet, implementeret og testet spillet Monopoly Junior

Monopoly Junior omfatter hele 2-4 spillere, som hver især kæmper om ikke at gå fallit og vinde ved at have flest mulige penge. For at danne sig en realistisk visualisering af spillets komposition, er der blandt andet gjort brug af en domænemodel, sekvensdiagram, design diagram og en use case model. Herefter er dette blevet implementeret i Github, hvor spillet Monopoly Junior understøttes af flere former for under versioner: Feature\_1 og Chance\_card. I disse features implementeres de pågældende krav, som blandt andet bliver nuanceret under “Krav og analyse”. Monopoly Junior bliver forhånds sikret ved hjælp af diverse tests. Her bliver der blandt andet lavet en Junit test, brugertest og en test af spillet ved 100, 1.000, 10.000 og 100.000 terningekast. Dette bliver også forklaret i detaljer under “Test”.

#### **5.Projektplanlægning**

Denne risikoanalyse viser de mest sandsynlige risici under projektets forløb. hver risiko har et nummer for “impact” og “probability” som ganget sammen viser et nummer for hvor stor betydning risikoen har. Dvs. risici skal tages i betragtning først hvis deres nummer er højt.

Nr	Risk	Probability	Impact	P*I
1	for kort tidsramme	3	3	9
2	Udefinerede krav	3	2	6
3	mangel af noter for forståelse	2	3	6
4	Overskride krav for projekt	1	2	2
5	manglende erfaring i programmering	2	1	2
		4 high - 1 low	4 high - 1 low	Probability * Impact

	Impact				
Probability		1	2	3	4
	1		nr. 4		
	2	nr. 5		nr. 3	
	3		nr. 2	nr. 1	
	4				

*Figur 5.1 Risikoanalyse*

hver risici har sin egen placering på figur 5.1. hvis man tager udgangspunkt i risiko 2 om udefinerede krav, kan man se at den har en sandsynlighed på 3 og en impact på 2, dvs. risikoen er relativ høj og den skal tages i betragtning og være i bagtankerne under projektets forløb. der er dog kun en risiko under den orange del af figuren, hvilket er den mest sandsynlige risiko, som forhåbentligt kan overkommes. risiko nr 5 og 4 har mindst sandsynlighed og impact, og er derfor placeret på det grønne felt af figuren.

### Plan for risici

Risiko **nr 1**. regnes med at være det mest sandsynlige problem, derfor har vi valgt at lave en prioriteret liste som ligger under **6.krav/analyse** for netop at tage højde for dette. risiko **nr 2**. ender også med at få nytte af denne liste.

Risiko **nr 3**. som handler om forståelse af hinandens kode vha. noter, har vi valgt at prøve at løse vha. samtaler om individuelle bidrag til koden, og til dels ved at tilføje lidt ekstra noter her og der i koden.

## 6.Krav/Analyse

Krav for projektet er opstillet fra kundens vision, hvilket har ændret sig løbende, p.g.a. dialoger med kunden. nedenfor ses kundens krav fra starten af projektet uden supplerende samtaler og diskussioner.

## Kundens Vision:

*I skal udvikle et **Monopoly Junior** spil. Vurder hvad der er det vigtigste for at spillet kan spilles! Implementer de væsentligste elementer for at spillet kan spilles. I må **gerne** udelade regler - prioritér!*

[Spilleregler på dansk](#)

[Spilleregler på engelsk](#)

[Spilleplade](#)

[Chancekort](#)

*Nu har vi terninger og spillere på plads, men felterne mangler stadig en del arbejde. I dette tredje spil ønsker vi derfor at forrige del bliver udbygget med forskellige typer af felter, samt en decideret spilleplade.*

*Spillerne skal altså kunne lande på et felt og så fortsætte derfra på næste slag. Man går i ring på brættet.*

*Der skal nu være 2-4 spillere.*

## Vores tolkning af kravene.

Hver spiller optræder med eget navn og får tildelt en rolle (Bil, Skib, Hund eller Kat). Rollen er nødvendig for at 4 chancekort kan bruges som beskrevet.

Det er ikke klart skrevet nogen steder at der kun skal være en terning, men vores erfaring er at spilleren kommer for hurtigt rundt på brættet hvis der 2. Så vi har kun en.

Spillerens køretøj på brættet er i vores spil en rød personbil, gul sportsvogn, blå traktor eller grøn flyvende tallerken

## Vores prioriterede liste af krav, som programmet skal opfylde.

1. Felterne skal kunne vises på brættet.
2. Spilleren skal kunne slå med en terning.
3. Bank funktionen skal fungere. dvs. spilleren får penge for at passere start bliver trukket når det er specificeret(køb af ejendom, afgift osv.)
4. bilerne skal flyttes rundt på brættet som konsekvens af terningslaget.
5. Spillerne skal kunne købe felter.
6. De 20 chancekort skal kunne trækkes når spilleren lande på et chance felt.
7. Den aktion chance kortet foreskriver skal udføres.
8. Spilleren skal kunne komme i fængsel.
9. Betaling af dobbelt husleje.
10. Spillet skal kunne gentages.
11. Implementering af avanceret spillemåde. Det betyder at du kan betale med ejendomme du ejer

## Kravspecifikation:

### Functionality:

Programmet skal fungere i java 18.  
programmet skal kunne konverteres til en jar fil.  
Spilles af 2 til 4 spillere

### **Usability:**

Vis alle spilleres score altid.  
Se hvis tur det er  
Felter kan købes  
Felter har forskellige farver  
Vis felt der landes på  
Vis terninger  
Vis brikker/roller

### **Reliability:**

Spillet skal kunne spille mindst et spil uden at vise en fejlbesked

### **Performance:**

-

### **Supportability:**

skal kunne laves i forskellige sprog

GRASP patterns -

nedenfor beskrives hvilke former for klasser er brugt i projektet, ud fra GRASP patterns.

### **Information expert**

Base klassen indeholder information som bruges i næsten alle andre klasser som fx. mængden af felter, mængden af spillere og pris for hvert felt.

### **Creator**

BoardCreator klassen danner de fleste ting som har med selve spillepladen at gøre, her bliver der dannet en plade array som indeholder forskellige linjer fra text filer.

### **Controller**

Fields klassen under mappen GameMechanics har nogle aspekter af en controller, når den fx. skal flytte bilerne rundt på pladen hvor den samler nogle metoder for at slette og tilføje biler til pladens felter.

### **Indirection**

Det bedste eksempel på indirection er nok textReaderClass under GameMechanics, som simplificerer det at læse en textfil så der ikke skal laves et nyt system hver gang der skal læses fra en text fil.

### **High Cohesion**

### **Low Coupling**

Base Klassen under TheBoard benyttes til at forsøge at lave high cohesion og low coupling, ved at lave nogle universelle værdier.

### **Polymorphism**

Chance Klassen under cardClasses har en del polymorphism i form af arv fra alle andre klasser i cardClasses mappen.

### **Protected variations**

indtil videre er der ikke nogle protected variations i programmet.

### **Pure fabrication**

Der er ikke en klasse baseret på pure fabrication, dog er der nogle aspekter i Fields klassen under GameMechanics, som kan klassificeres som pure fabrication. Disse aspekter er hhv. ColorSpace metoden som sætter farver på felter.

## **7.Versionering**

### **Versionsstyring**

Lav et lokal Git-repository som en del af IntelliJ-projektet.

Alternativt kan afleveres et link til et repository på nettet eks.

<https://github.com/cbudtz/TestRepo42.git>.

Rapporten skal indeholde en vejledning i hvordan man importerer Git-repository i IntelliJ

### **Krav og analyse:**

Der skal udarbejdes følgende artifacts:

- Kravliste
- Use case diagram
- Eksempler på use case beskrivelser - vælg mindst én, der beskrives fully dressed
- Domænemodel
- Et eksempel på systemsekvensdiagram
- Et eksempel på sekvensdiagram
- Design-klassediagram



## Konfigurationsstyring

Udviklingsplatformen er alt det software i bruger under udviklingen af jeres projekt. Produktionsplatformen er alt det software der skal bruges til at køre jeres færdige program. I dette projekt er de ens. I skal dokumentere platformens dele med versionsnummer så den kan genskabes til senere brug. Jeres udviklingsplatform består af operativsystem, java, og IntelliJ samt biblioteket matadorgui.jar der hentes fra Maven.

I bedes definere hvordan I vil sikre jer, at I på et hvert tidspunkt vil kunne finde den sidste nye version af samtlige artefakter OG hvordan I vil sikre jer at dokumentationen altid er opdateret for jeres system

Øvrige bemærkninger:

### Spørgsmål til kunden:

- 1. Hvordan finder vi den nyeste version?
- 2. Hvordan sikrer vi at vi ved om versionen er opdateret?
- 3. Hvordan finder vi versioner som passer sammen?

– **Hvor er filerne?** Filerne findes under brugerens Git-repository. Før at brugeren kan få adgang til disse, skal vedkommende have modtaget en URL (<https://github.com/UffeBC/CDIOdel3Gruppe1.java>) med en invitation til det lokale Git-repository for projektet.

– **Hvordan finder vi den nyeste version?**

De nyeste versioner af filerne findes ved at opdatere den pågældende fil, dette sker ved at redaktionen pusher og comitter en opdatering til filen, hvilket medfører at personen der skal se opdateringen opdatere projektet får det færdigt.

– **Hvordan sikrer vi at vi ved om versionen er opdateret?**

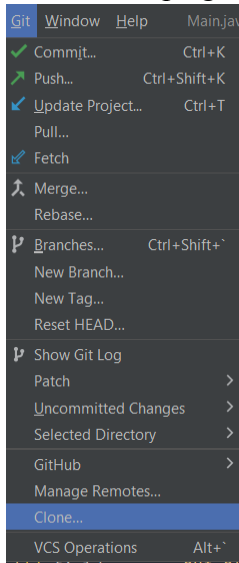
Dette ses øverst på vores main-class, hvor der er blevet tilføjet en kommentar ved begyndelsen af projektet (version 1.0), som vi så opdaterer hver gang at man har lavet et nyt commit i main.

– **Hvordan finder vi versioner som passer sammen?** For at sikre os at vi har arbejdet med versioner som passer sammen, har vi været inde på Maven repository og hentet en GUI. URL'en for denne har vi været inde og tilføje i pom.xml for vores projekt, for at sørge for at der bliver hentet den rette software, som en del af projektet. I tilfældet af at man opdatere til en nyere GUI, bør det tjekkes, om denne passer til programmets software.

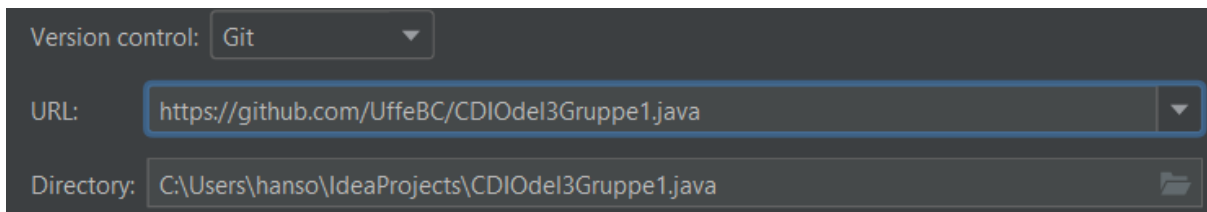
Endvidere skal i beskrive hvordan man importerer jeres projekt i IntelliJ fra git, og hvordan man kører jeres program.

1. Copy på linket: <https://github.com/UffeBC/CDIOdel3Gruppe1.java>
2. Åben intellij

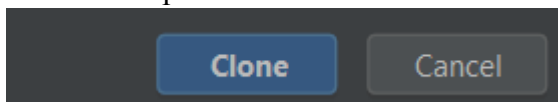
### 3. Klik på git > clone



### 4. Paste linket under URL



### 5. Klik på Clone nedenfor



Det er også et krav at I bruger Maven til at hente junit.

### Vores svar:

JUnit test er indbygget i IntelliJ IDE

Udviklingsplatform:

IntelliJ IDE 2022.2.1 (Ultimate Edition) Build #IU-222.3739.54, built on August 16, 2022

Programmet køres på Java 18

Java Developer Kit.

GUI bibliotek: <https://github.com/diplomit-dtu/MatadorGUIDe/tree/3.2.x>

# 8.Design

## Dokumentation

- Forklar hvad arv er.

### Vores svar:

En superklasse har de fælles metoder og attributter. Underklasserne arver disse har selv yderligere metoder og attributter der er specifikke for hver deres klasse. Desuden kan underklasserne overskrive super-klassens metoder.

- Forklar hvad abstract betyder.

### Vores svar:

En abstrakt klasse eller metode kan ikke instantieres.

- Fortæl hvad det hedder hvis alle field-klasserne har en landOnField metode der gør noget forskelligt.

### Vores svar:

Polymorfi.

- Dokumentation for test med screenshots.
- Dokumentation for overholdt GRASP.

## Use Case beskrivelser (Brief, den centrale use case i fullydressed)

### Use case UC1: Spil spillet

#### Primary actor: Spillerne

#### Stakeholders and interest:

**-Spillerne:** Spillerne forventer at spillet Monopoly Junior har en overensstemmelse med spillets regler og firmaets vision. Derudover skal spillets funktionalitet være fejlfrit.

**-Producenterne:** Producenterne har et ønske om at tilfredsstille kundens behov ud fra deres ønsker og krav. Som tilsammen danner rammerne for et hyggeligt og velfungerende spil.

**-Læreren:** Læreren vil have at producenter følger kundens krav og vision, samt udarbejder et fejlfrit spil som kan opfylde og tilfredsstille kundens og deres spilleres behov

**-Kunden:** Kunden forventer et funktionelt spil, som overholder deres krav. Herunder skal spillet kunne spilles med en minimal indsats og helt problemfrit

**Preconditions:** Der skal være en computer tilgængelig, som blandt andet har java 18. Før spillet skal være funktionelt skal der være et bræt med hele 24 felter, chancekort, 1 terning og 2-4 spillere

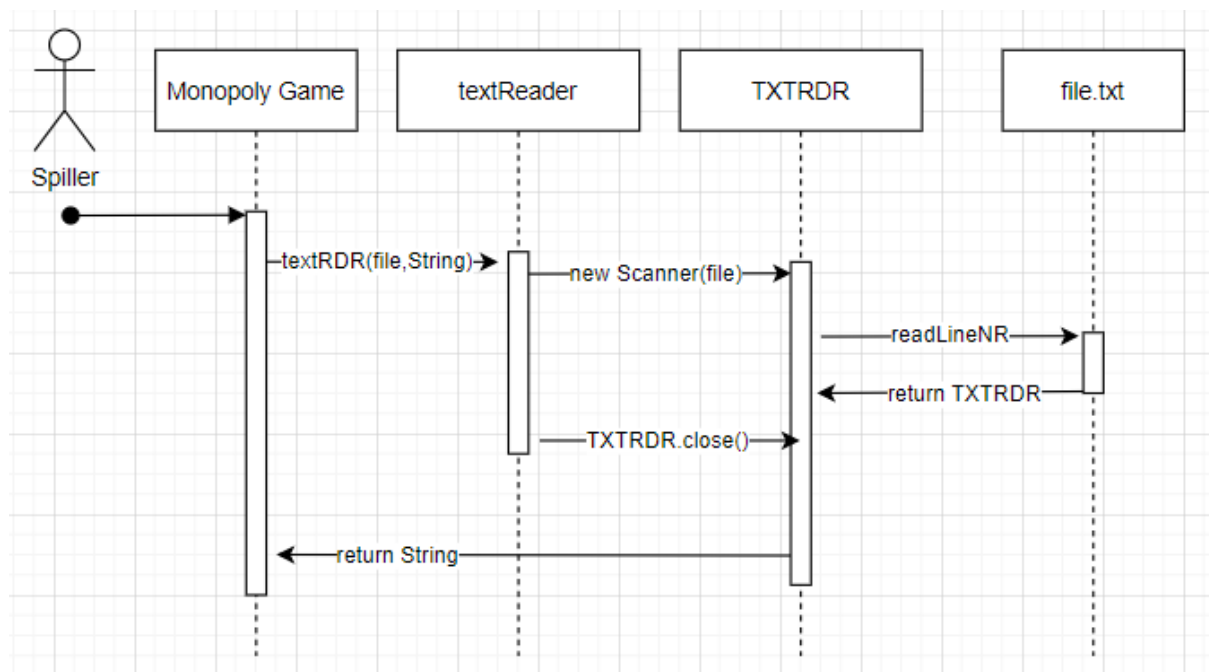
**Success Guarantee (Postconditions):** Spillerne har valgt det ønskede sprog. spillet er blevet afsluttet med en vinder, som har haft flest mulig penge i løbet af en sessions spil.

### Main Success Scenario (or Basic flow):

2-4 spillere er klar til at spille

1. 2-4 spillere er klar til at spille Monopoly Junior med en computer
2. Det ønskede sprog vælges af spillerne
3. Antal spillere indtastes
4. Spiller navne indtastes til identificering
5. Spillerne vælger hver deres rolle/brik
6. En af spillerne starter med at slå med terningen
7. Spillerne fortsætter indtil en spiller har opnået mest rigdom eller hvis en eller flere går fallit
8. En af spillerne vinder
9. Spillet lukkes ned eller genstartes

For at visualisere hvordan at vores system læser diverse filer i vores program, har vi opstillet et sekvensdiagram.



9.0 Sekvensdiagram for læser af .txt filer

Dette sekvensdiagram viser hvordan textRDR metoden læser og viser text fra en fil baseret på filens placering og linjen som der skal læses fra. På sekvensdiagrammet kan vi helt præcis se, hvordan at textRDR kalder på en scanner, hver gang at den selv bliver kaldt på. Denne scanner henvender sig så til den gældende fil, hvor at læser hver en linje og returnerer en String, hvor denne process så gentages indtil at filen er tom. Herefter termineres scanneren.  
*note: textRDR metoden ligger under textReaderClass klassen i programmet.*

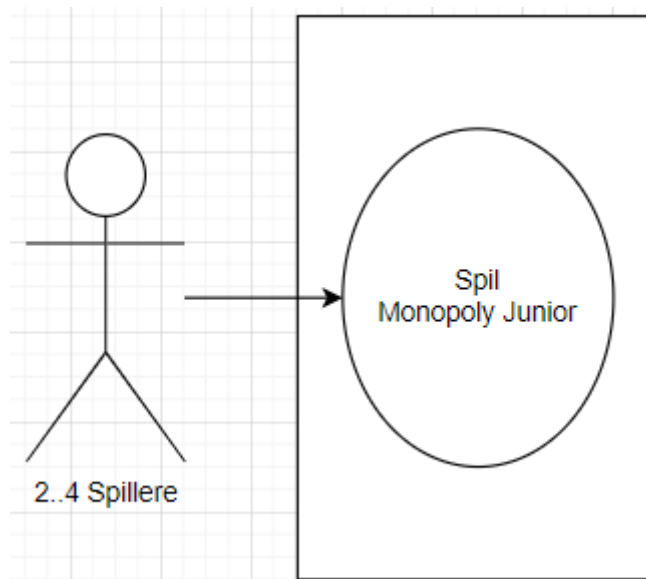
Da vi i dette program har en hel del chancekort af forskellige type, har det været oplagt for os at opstille disse i forskellige klasser. Disse arver dog en hel del fra klassen "Chancekort", som der indeholder den fundamentale kode for chancekortenes funktion. Som det kan ses herunder, har vi valgt at de forskellige chancekort, skal arve fra moderklassen. Dette illustreres ved følgende domænemodel:

## Analyse og designdokumentation

Kravliste:

- Brug af GUI
- Forskellige typer felter
- gå i ring på brættet
- 2-4 spillere
- Junior Monopoly spil

Diagrammer:



9.1 Use Case Diagram

I monopoly junior spillet er nogenlunde alt op til tilfældigheder, hvilket betyder at der ikke er nogen sub use-cases endnu. Vi kan dog se ved ovenstående figur, hvordan at den centrale use case "spil Monopoly Junior" er stillet op.

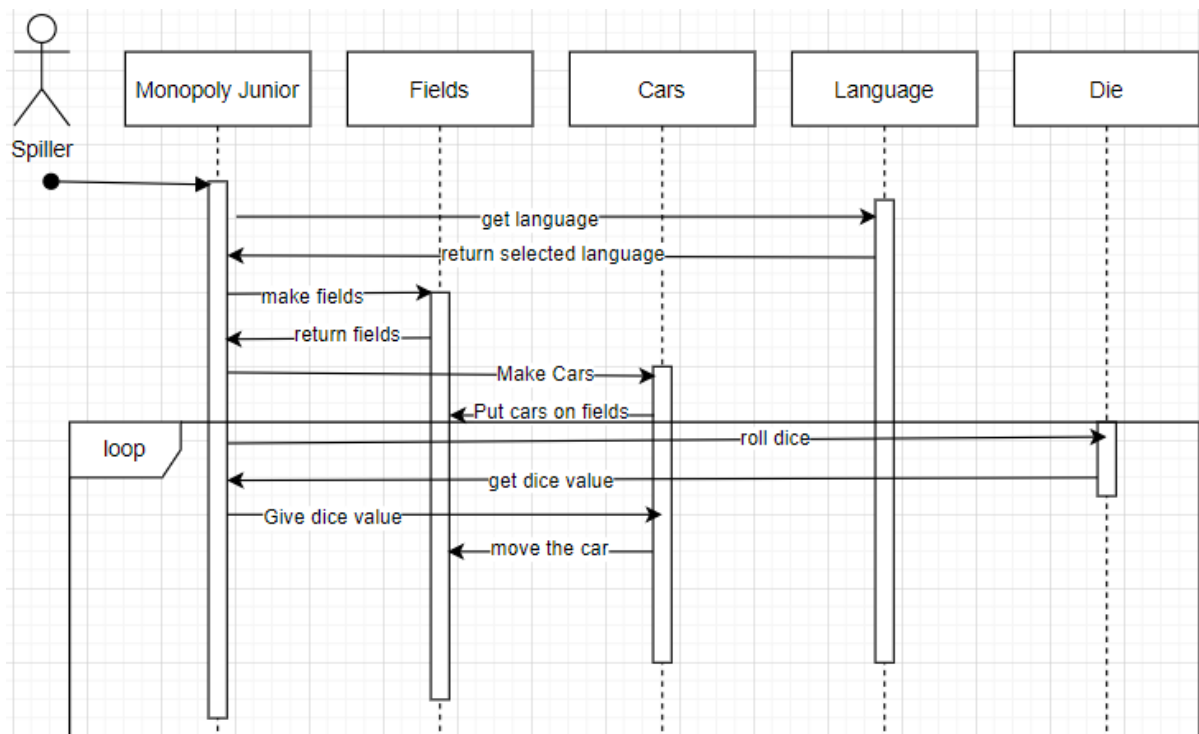
### - Eksempel på Use Case Beskrivelse Fully Dressed -

I Junior versionen af Matador er der ikke nogle konkrete usecases andet en at spille spillet. Hvis man skulle komme på et eksempel, kunne man se på et Matadorspil med 40 brikker hvor der er en funktion hvor spillere kan forveksle ejendomme ved at give forskellige bud på køb og salg. En sådan sub-use case beskrivelse fully dressed kunne se sådan ud:

#### **Sub-Use Case - Handel om Slikbutik**

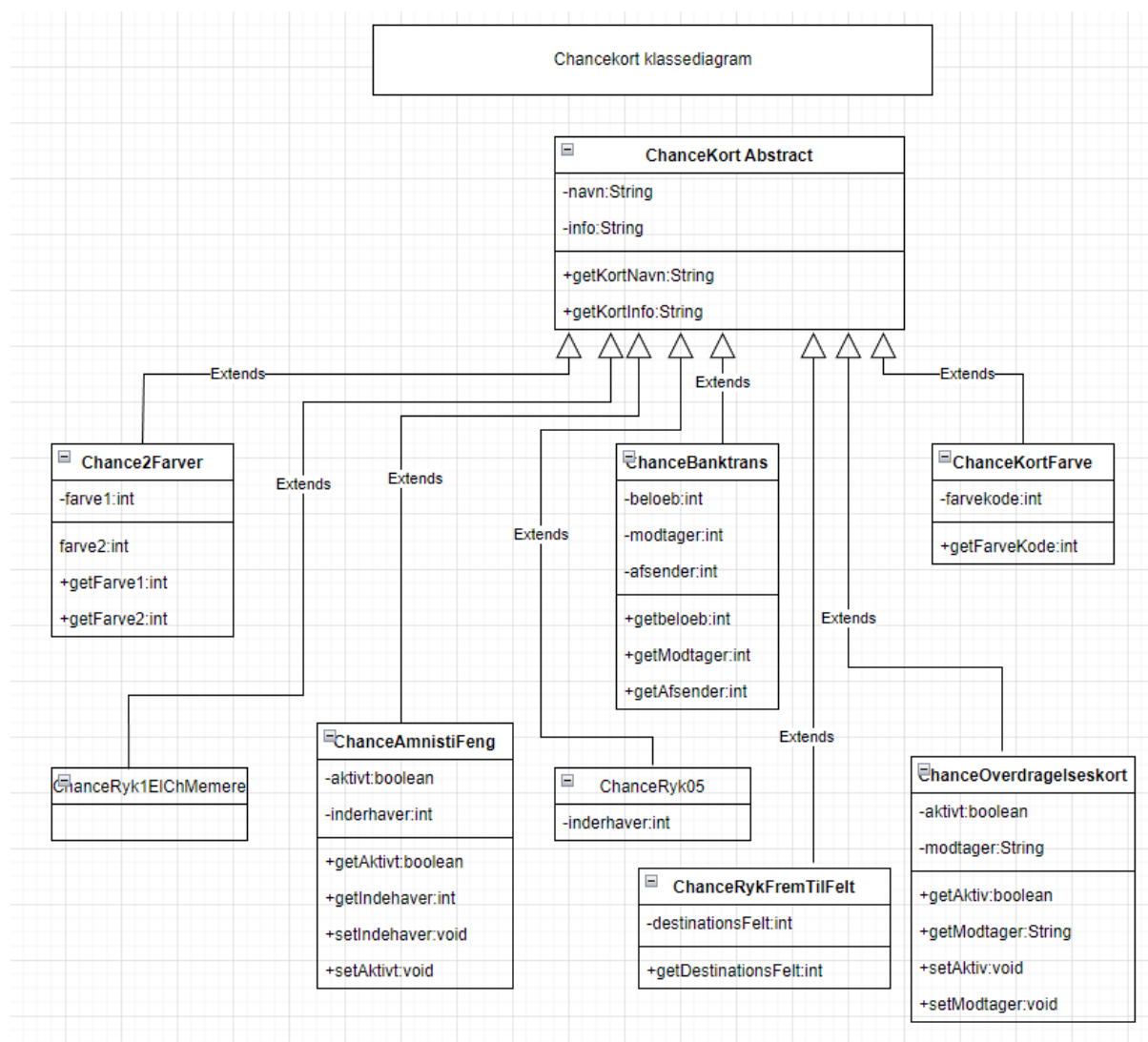
*Spiller1 og Spiller2 har besluttet at gå i forhandling om at Spiller 1 vil købe Spiller2's Slikbutik. Spiller1 starter med at give et bud på 200€, hvilket Spiller2 ikke er tilfreds med, hvorefter Spiller2 giver et tilbud på 450€. Spiller1 går på kompromi og giver et andet bud på 350€, dette tilfredsstiller Spiller2, og handlen går igennem. Spiller1 ejer nu Slikbutikken og er 350€ fattigere, og Spiller2 har tjænt 350€ og solgt sin Slikbutik.*





#### 9.4 Systemsekvensdiagram

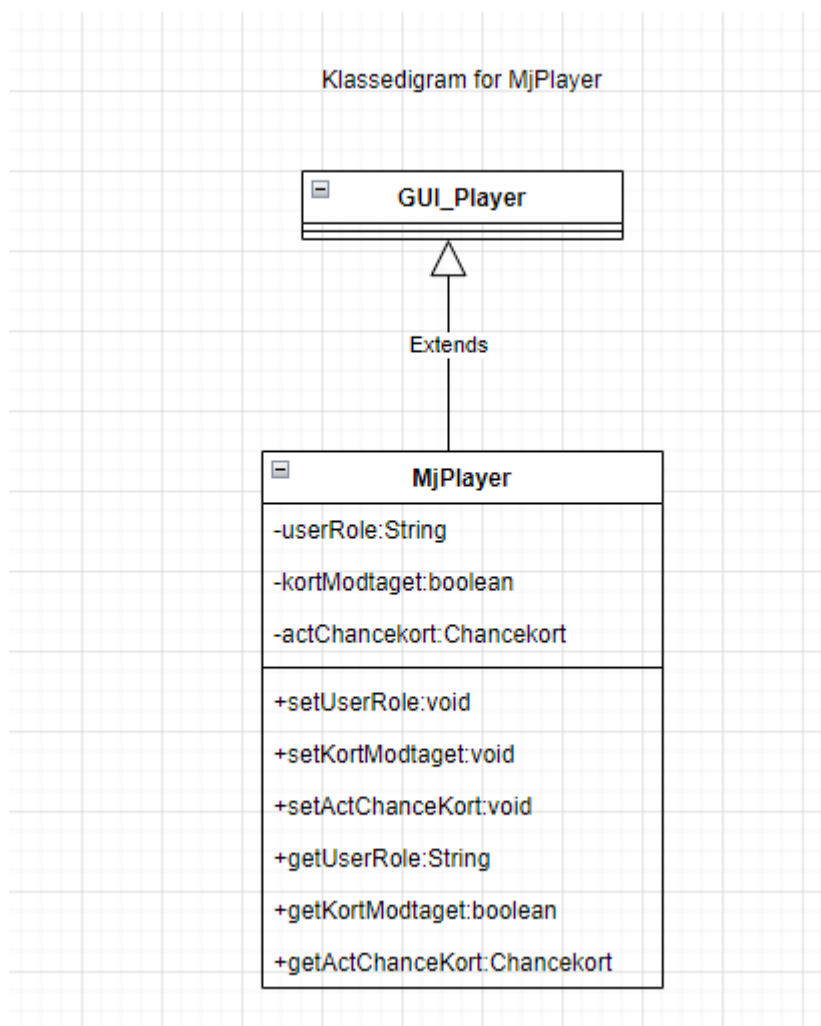
Dette diagram 9.3 viser hvilke dele af systemet som interagerer med hinanden, når spillet kører. Når spillet køres vil der blive valgt et sprog, hvorefter felterne og bilerne dannes. Nu begynder spillet konkret at begynde da den første spiller slår terningen og flytter bilen som er vist med et loop da det gentages for hver spiller efter hinanden. Når en person har 0 point tilbage, findes der en vinder og spillet slutter.



9.5 for chancekort - designklassediagram (pil op til klasse for "spil")

figur 9.5 skaber et billede af, hvordan klasserne for de forskellige chancekort, allesammen arver et sæt grundlæggende funktioner fra moderklassen "Chancekort Abstract". Derudover ser vi nogle detaljer for, hvordan hver klasse er opbygget og hvilke nøglerlementer/funktioner disse indeholder, som ikke er en del af nedarvningen fra moderklassen.





9.6 klassediagram for klassen MjPlayer.

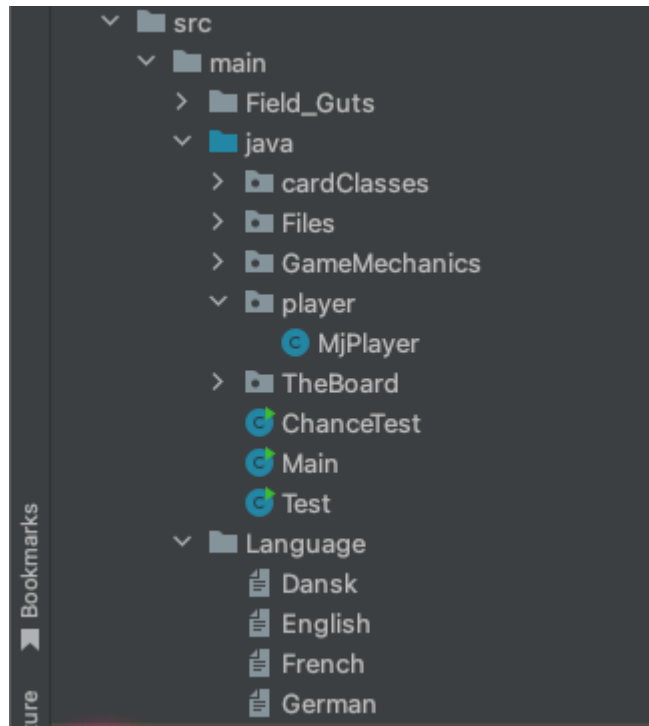
Dette diagram viser, hvordan at klassen MjPlayer, nedarver visse funktioner fra klassen GUI\_player, som der er blevet nedarvet fra maven. Derudover ser vi en række Get-Set metoder, der sørger for at oplysningerne for den enkelte spiller er på plads.

## 9.Implementering

### Implementering

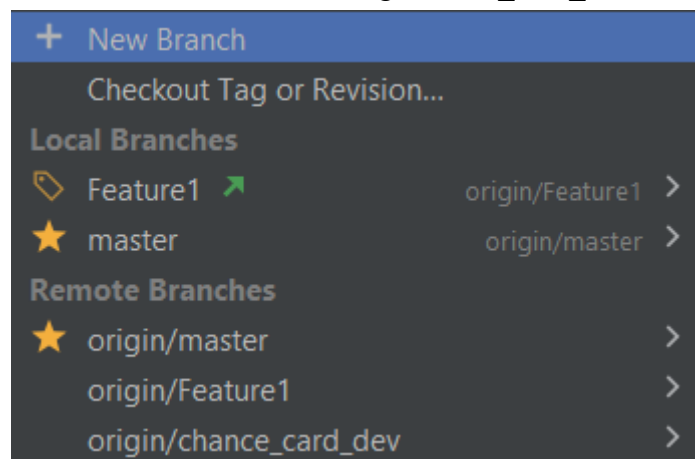
- Lav passende konstruktører.
- Lav passende get og set metoder.
- Lav passende toString metoder.
- Lav en klasse GameBoard der kan indeholde alle felterne i et array.
- Alt skal udvikles efter Objektorienterede principper, hvilket for felternes vedkommende vil sige at de skal laves med et fornuftigt arvehierarki.

- Tilføj en toString metode der udskriver alle felterne i arrayet.
- Lav det spil kunden har bedt om med de klasser I nu har.
- Benyt GUI'en. Gui' skal importeres fra Maven: [Maven repository](#)



*Figur 10.0 overblik over klasserne*

Under programmeringen er der brugt versionering fra git, hvor matadorspillet er blevet spredt ud over 3 generelle dele. hhv. master, Feature1 og Chance\_card\_dev.



*Figur 10.1 programmets opstilling*

Programmets forskellige grene har haft forskellige arbejdsopgaver, her er brættets opstilling og genert spilmekanik lavet i Feature1, og chance kort som er dannet i chance\_card\_dev.

Disse klasser samles i master hvor det generelle program skal kører.

Grunden til at bruge disse forskellige grene, er for at undgå konflikter, og for altid at have et projekt som virker i

## 10.Test

*Lav mindst tre testcases med tilhørende fremgangsmåde/testprocedure og testrapporter.  
Lav mindst én Junit test til centrale metoder. Inkluder code coverage dokumentation.  
Lav mindst én brugertest. Husk at brugeren skal være en der ikke kan kode.  
Mange tests er løbende udført under udviklingen ved at printe variable ud til terminal vinduet.*

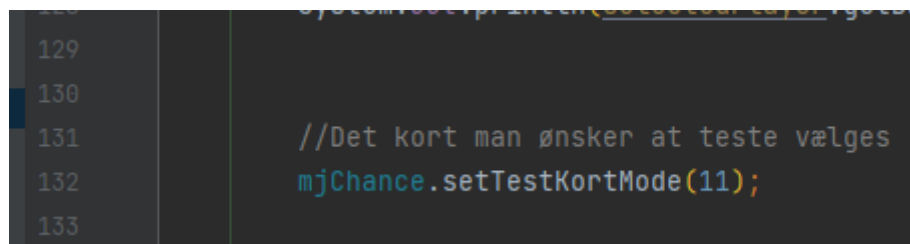
### BrugerTest

I slutningen af projektet har vi lavet en brugertest med en potentiel kunde som ikke har programmeringserfaring. Kunden klikkede sig frem i spillet uden problemer, men var forvirret om hvad der egentlig foregik på spillepladen, og på hvordan spillet tællede ens score op eller ned. Efter at have spillet spillet var kunden dog tilfreds med at have vundet over deres modstander og valgte at prøve spillet en gang til.

Brugerens kritik var at der generelt manglede noget overblik i form af et sæt regler, eller nogle flere pop-ups som forklarer forløbet.

### Test af Chancekort

Chance kortene er systematisk testet med klassen “ChanceTest”. Det er en Junit test af klassen “Chance”. “ChanceTest” indeholder en main og kan køres selvstændigt. Inden test sættes bræt, spillere og køretøjer. I linje 132 vælger man hvilket kort, der skal testes:

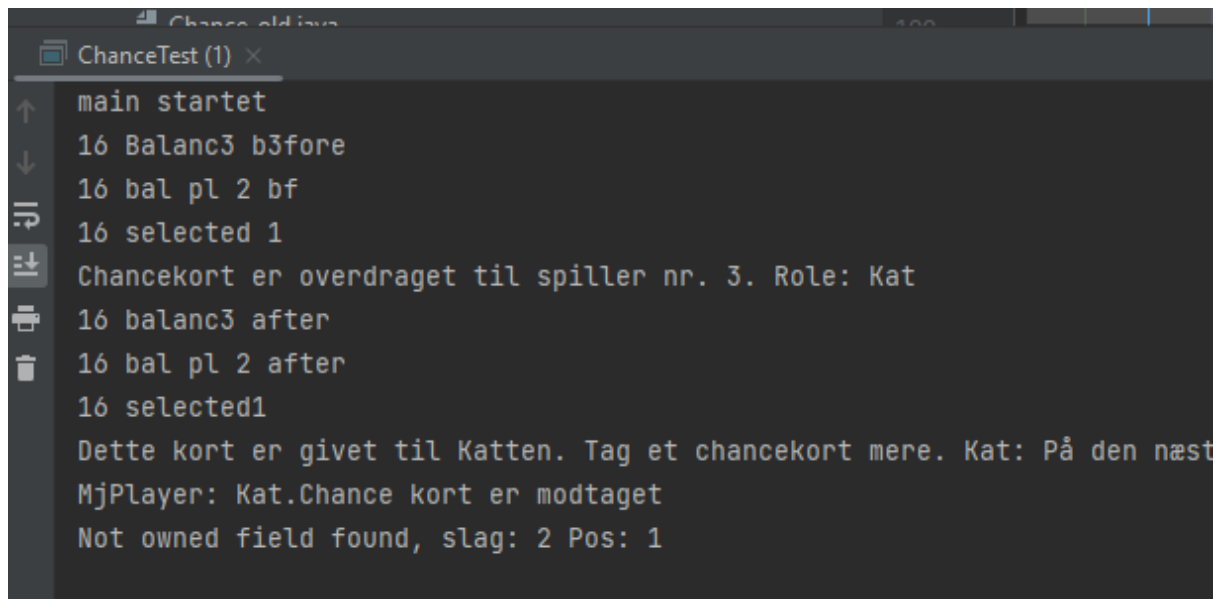


```
129  
130  
131 //Det kort man ønsker at teste vælges  
132 mjChance.setTestKortMode(11);  
133
```

*Screenshot 10.1 - Test af chance kort nr 11.*

I dette tilfælde er kort nr. 11 valgt.

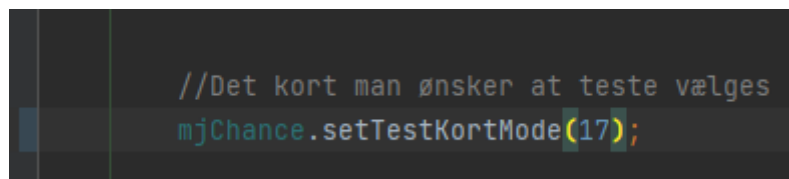
Første slag rammer første chance felt.



```
main startet
16 Balanc3 b3fore
16 bal pl 2 bf
16 selected 1
Chancekort er overdraget til spiller nr. 3. Role: Kat
16 balanc3 after
16 bal pl 2 after
16 selected1
Dette kort er givet til Katten. Tag et chancekort mere. Kat: På den næst
MjPlayer: Kat.Chance kort er modtaget
Not owned field found, slag: 2 Pos: 1
```

*Screenshot 10.2 - Output af ChanceTest*

### Test af kort 17:



```
//Det kort man ønsker at teste vælges
mjChance.setTestKortMode(17);
```

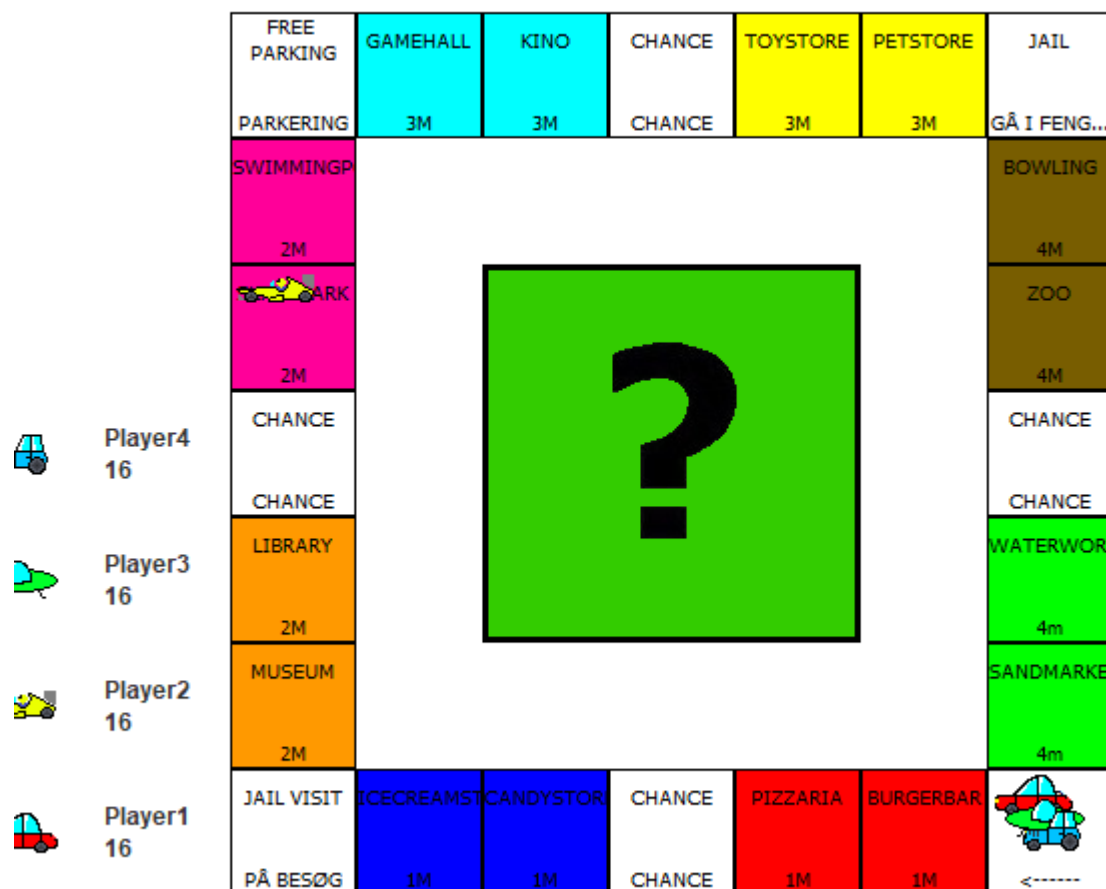
*Screenshot 10.3 - Test af Chancekort*

Gratis felt. Du Rykkes frem til Skaterparken for at lave det perfekte grind. Hvis ingen ejer den, får du den gratis. Ellers skal du betale leje ejeren.

OK

		FREE PARKING	GAMEHALL	KINO	CHANCE	TOYSTORE	PETSTORE	JAIL
		PARKERING	3M	3M	CHANCE	3M	3M	GÅ I FENG...
		SWIMMINGP						BOWLING
		2M						4M
		SKATEPARK						ZOO
		2M						4M
		CHANCE						CHANCE
		CHANCE						CHANCE
	Player4 16	LIBRARY						WATERWORK
	Player3 16	2M						4m
	Player2 16	MUSEUM						SANDMARKE
		2M						4m
	Player1 16	JAIL VISIT	ICECREAMST	CANDYSTOR	 CHANCE	PIZZARIA	BURGERBAR	
		PÅ BESØG	1M	1M	CHANCE	1M	1M	<-----

*Screenshot 10.4 - Der klikkes ok og det ses på screenshot 10.3 at bilen er rykket frem Skaterparken*



Screenshot 10.5 - Bil rykkes

Test af kort 17:

```
//Det kort man ønsker at teste vælges
mjChance.setTestKortMode(3);
```








Screenshot 10.6 Test af chancekort nr 3.

Gratis felt. Du RykKes frem til et orange felt. Hvis det der ledigt, får du det gratis. Ellers skal du betale leje ejeren.

OK



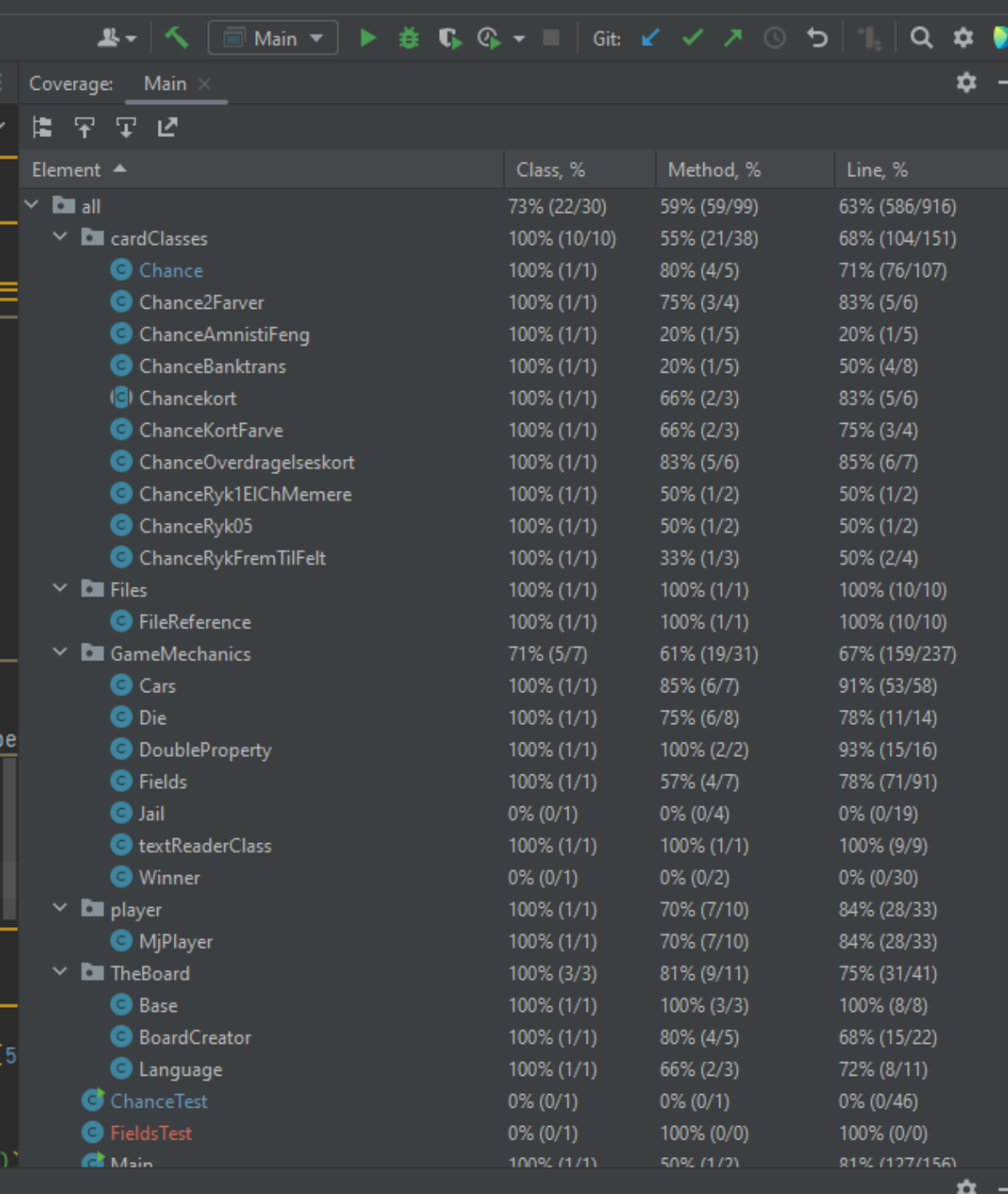
Screenshot 10.7 - Der klikkes ok og derefter på screenshot 10.7 ses at bilen er rykket frem til et orange felt.

		FREE PARKING	GAMEHALL	KINO	CHANCE	TOYSTORE	PETSTORE	JAIL
		PARKERING	3M	3M	CHANCE	3M	3M	GÅ I FENG...
		SWIMMINGP						BOWLING
		2M						4M
		SKATEPARK						ZOO
		2M						4M
		CHANCE						CHANCE
		CHANCE						CHANCE
		LIBRARY						WATERWORKS
		2M						4m
		 2M	SANDMARKET					
		2M	4m					
		JAIL VISIT	ICECREAMST	CANDYSTOR	CHANCE	PIZZARIA	BURGERBAR	
		PÅ BESØG	1M	1M	CHANCE	1M	1M	<-----
	Player4 16							
	Player3 16							
	Player2 16							
	Player1 16							

Screenshot 10.8 Bil rykkes



**Et spil, spillet til ende, med coverage:**



Element	Class, %	Method, %	Line, %
all	73% (22/30)	59% (59/99)	63% (586/916)
cardClasses	100% (10/10)	55% (21/38)	68% (104/151)
Chance	100% (1/1)	80% (4/5)	71% (76/107)
Chance2Farver	100% (1/1)	75% (3/4)	83% (5/6)
ChanceAmnistiFeng	100% (1/1)	20% (1/5)	20% (1/5)
ChanceBanktrans	100% (1/1)	20% (1/5)	50% (4/8)
Chancekort	100% (1/1)	66% (2/3)	83% (5/6)
ChanceKortFarve	100% (1/1)	66% (2/3)	75% (3/4)
ChanceOverdragelseskort	100% (1/1)	83% (5/6)	85% (6/7)
ChanceRyk1ElChMemere	100% (1/1)	50% (1/2)	50% (1/2)
ChanceRyk05	100% (1/1)	50% (1/2)	50% (1/2)
ChanceRykFremTilFelt	100% (1/1)	33% (1/3)	50% (2/4)
Files	100% (1/1)	100% (1/1)	100% (10/10)
FileReference	100% (1/1)	100% (1/1)	100% (10/10)
GameMechanics	71% (5/7)	61% (19/31)	67% (159/237)
Cars	100% (1/1)	85% (6/7)	91% (53/58)
Die	100% (1/1)	75% (6/8)	78% (11/14)
DoubleProperty	100% (1/1)	100% (2/2)	93% (15/16)
Fields	100% (1/1)	57% (4/7)	78% (71/91)
Jail	0% (0/1)	0% (0/4)	0% (0/19)
textReaderClass	100% (1/1)	100% (1/1)	100% (9/9)
Winner	0% (0/1)	0% (0/2)	0% (0/30)
player	100% (1/1)	70% (7/10)	84% (28/33)
MjPlayer	100% (1/1)	70% (7/10)	84% (28/33)
TheBoard	100% (3/3)	81% (9/11)	75% (31/41)
Base	100% (1/1)	100% (3/3)	100% (8/8)
BoardCreator	100% (1/1)	80% (4/5)	68% (15/22)
Language	100% (1/1)	66% (2/3)	72% (8/11)
ChanceTest	0% (0/1)	0% (0/1)	0% (0/46)
FieldsTest	0% (0/1)	100% (0/0)	100% (0/0)
Main	100% (1/1)	50% (1/2)	81% (127/156)

*Screenshot 10.9 - Overblik over coverage af klasser*

## Test af terninger.

Vi bruger samme terninger som i CDIO 01. Se litteraturliste punkt 4. Testen i CDIO 01 viste at terningerne fungerer som ønsket.

## 11.Konklusion

Vi har et velfungerende spil med mange funktioner implementeret. Se venligst vores prioriteringsliste under afsnit 6. 9 ud af 10 krav er implementeret og spillet virker velfungerende ved afprøvning. Ud fra kravspecifikationerne har vi fået lavet en hel del af det vi forventede kunne laves indenfor den givne tidsramme. Vi har haft meget nytte af vores risikoanalyse, da vi netop endte med at have problemer med tidsrammen. Designet passer til programmeringen, med få afvigelser I form af små ændringer i metode navne og måderne som metoder fungere, men med det samme funktionalitet.

Der hvor vores program kunne forbedres er specielt I form af forklaringer, som måske godt kunne have blevet prioriteret lidt mere, da det kan være svært at følge med i hvem der ejer hvilke felter og hvorfor spillernes pengebeholdning ændrer sig som den gør, dette er dog blevet forbedret ved at tilføje flere forklaringer. når det kommer til sprog, har vi kun nået at tilføje sprog til knapper med funktionaliteter som fx. "kast terning", "vælg sprog" og "vælg rolle".

Der er selvfølgelig altid flere funktionaliteter man kan tilføje til programmet, men inden for den givne tidsramme og med den relativt lille erfaring, stiller vi os tilfredse med vores projekt.

## 12.Bilag

1	Til	Chancekort mere		Udfordring	Destination	Aktion	Class
2	Bilen	Ja	Bil: På din næste tur skal du drene frem til et hvilket som helst ledigt felt og købe det. Hvis det ikke er nogen ledige felter skal du købe et fra en anden spiller!	Kortet skal overdrages til anden spiller. Denne kan vælge et hvilket som helst felt. Hvis ikke ledigt felt hvordan besluttet hvilket felt der købes?	Bilen	Overdrag	ChanceOverdragelseskort
3	den aktive spiller	Nej	Ryk frem til start.	Konsekvens ikke veldefineret. Spilleren kan antagelig vælge.	Start	ryk	ChanceRykFremTilFelt
4	den aktive spiller	Nej	Ryk op til 5 felter frem		0/1...5 felter	ryk	ChanceRyk05
5	den aktive spiller	Nej	Gratis felt. Ryk frem til et orange felt. Hvis det der ledigt, får du det gratis. Ellers skal du betale leje ejeren.		Orange. Kode 3	ryk	ChanceKortFarve
6	den aktive spiller	?	Ryk 1 felt frem eller tag et chancekort mere	Spilleren har et valg	1 felt	ryk	ChanceRyk1ElChMemere
7	Skibet	Ja	Giv dette kort til skibet og tag et chancekort mere. Skib: På den næste skal du sælle frem til hvilket som helst ledigt felt og købe det. Hvis der ikke er nogen ledige felter, skal du købe et af en anden spiller.	Kortet skal overdrages til anden spiller. Denne kan vælge et hvilket som helst felt. Hvis ikke ledigt felt hvordan besluttet hvilket felt der købes?	Skibet	Overdrag	ChanceOverdragelseskort
8			Du har spist for meget slik. Betal M2 til banken.		Betal	pengetransaktion med Banken	ChanceBanktrans
9	den aktive spiller	Nej	Gratis felt. Ryk frem til et orange eller grønt felt. Hvis det der ledigt, får du det gratis. Ellers skal du betale leje ejeren.	Spilleren har et valg	Orange eller grønt. Kode 3 og 8	ryk	Chance2Farver
10	den aktive spiller	Nej	Gratis felt. Ryk frem til et lyseblåt felt. Hvis det der ledigt, får du det gratis. Ellers skal du betale leje ejeren.		Lyseblåt. Kode 5	ryk	ChanceKortFarve
11	den aktive spiller	Nej	Du læslades uden omkostninger. Behold dette kort til du får brug for det.	Spilleren beholder kortet		Amnesti fængsel	ChanceAmnestiFeng
12	den aktive spiller	Nej	Ryk frem til Strandpromenaden.		Strandpromenaden. Felt 23	ryk	ChanceRykFremTilFelt
13	Katten	Ja	Giv dette kort til Katten og tag et chancekort mere. Kat: På den næste skal du sælle frem til hvilket som helst ledigt felt og købe det. Hvis der ikke er nogen ledige felter, skal du købe et af en anden spiller.	Kortet skal overdrages til anden spiller. Denne kan vælge et hvilket som helst felt. Hvis ikke ledigt felt hvordan besluttet hvilket felt der købes?	Katten	Overdrag	ChanceOverdragelseskort
14	Hunden	Ja	Giv dette kort til Hunden og tag et chancekort mere. Hund: På den næste skal du sælle frem til hvilket som helst ledigt felt og købe det. Hvis der ikke er nogen ledige felter, skal du købe et af en anden spiller.	Kortet skal overdrages til anden spiller. Denne kan vælge et hvilket som helst felt. Hvis ikke ledigt felt hvordan besluttet hvilket felt der købes?	Hunden	Overdrag	ChanceOverdragelseskort
15	den aktive spiller	Nej	Det er din fødselsdag. Alle giver dig 1 M. Tillykke med fødselsdagen		Modtag	pengetransaktion med de andre deltagere	ChanceBanktrans
16	den aktive spiller	Nej	Gratis felt. Ryk frem til et pink eller mørkeblåt felt. Hvis det der ledigt, får du det gratis. Ellers skal du betale leje ejeren.	Spilleren har et valg	Pink eller Mørkeblåt.Kode 4 eller 2	ryk	Chance2Farver
17			Du har lavet alle dine lektier. Modtag 2M fra banken		Modtag	pengetransaktion med Banken	ChanceBanktrans
18	den aktive spiller	Nej	Gratis felt. Ryk frem til et rødt felt. Hvis det der ledigt, får du det gratis. Ellers skal du betale leje ejeren.		Rødt. Kode 1	ryk	ChanceKortFarve
19	den aktive spiller	Nej	Gratis felt. Ryk frem til Skaterparken for at lave det perfekte grind. Hvis ingen ejer den, får du den gratis. Ellers skal du betale leje ejeren.		Skaterparken. Felt 10	ryk	ChanceRykFremTilFelt
20	den aktive spiller	Nej	Gratis felt. Ryk frem til et lyseblåt eller rødt felt. Hvis det der ledigt, får du det gratis. Ellers skal du betale leje ejeren.	Spilleren har et valg	Lyseblåt eller Rødt.Kode 5 eller 1	ryk	Chance2Farver
21	den aktive spiller	Nej	Gratis felt. Ryk frem til et brunt eller gult felt. Hvis det der ledigt, får du det gratis. Ellers skal du betale leje ejeren.	Spilleren har et valg	Brunt eller Gult.kode 9 eller 6	ryk	Chance2Farver
22							

### Forklaring af Farvekode:

**Farvekode**

**Farve**

**0**

**Hvid**

<b>1</b>	<b>Rød</b>
<b>2</b>	<b>Blå</b>
<b>3</b>	<b>Orange</b>
<b>4</b>	<b>Pink</b>
<b>5</b>	<b>Lyseblå</b>
<b>6</b>	<b>Gul</b>
<b>7</b>	<b>Brun</b>
<b>8</b>	<b>Grøn</b>

## Litteratur

1. Craig Larman: Applying UML and Patterns.
2. Git-Book.
3. Pearson: Lewis and Loftus: Java Software Solutions Foundations of Program Design.
4. CDIO del1: Torben Storm Rasmussen s047560, Tobias Schønau s224327,Uffe Clausen, s224275, Nabil Abbasi s224320, Ilias Elouali s221136
5. CDIO del2: Torben Storm Rasmussen s047560,Uffe Clausen, s224275, Nabil Abbasi s224320, Ilias Elouali s221136
6. Marven library