

02312

INDLEDENDE PROGRAMMERING

3 UGERS PROJEKT

Aya Fahim Yousif
S205466



Sheba Pirani
S205476

Mikail Oguzhan Kocak
S195205



Alec Ranjitkar
S205427

Yazan Mahmoud Bayazid
S181539



Jawahir Farah Shahal
S205414

Gruppe 18
18. januar 2021

GitHub Repo: https://github.com/Ayafahim/18_Matador.git

TIMEREGNSKAB

Studerende	S205466	S195205	S205476	S205427	S205414	S181539
Antal timer	50	50	50	50	50	50

ABSTRACT

This paper revolves around a project with the purpose of developing a software for a game called Matador for a customer who wants the game to be played at the databars on DTU. The game is intended as a digital board game, which can be played by two to six players. The players are marked as a car and each take turns in rolling the two six-sided die to move their car around the game board. The game board consists of 40 fields.

The purpose of the game is to drive one's opponents into bankruptcy by buying the different properties and collecting rent. The game ends when one player cannot pay the acquired rent.

To succeed in the project all the customer's requirements have been met. Furthermore, the game has been tested thoroughly to ensure the game's usability.

INDHOLDSFORTEGNELSE

Timeregnskab	2
Abstract	2
1 Indledning	4
2 Krav	5
3 Analyse	7
3.1 Use cases	7
3.2 Domain model	9
4 Design	10
4.1 Systemsekvensdiagram	10
4.2 Sekvensdiagram	11
4.3 Designklassediagram	11
5 Implementering	11
6 Test	14
7 Konfiguration styring	16
8 Mangler	17
9 Konklusion	18
10 GitHub repository	Fejl! Bogmærke er ikke defineret.
11 Bilag	18
12 Litteratur- og kildefortegnelser	40

1 INDLEDNING

I forbindelse med tidligere Design-Build-projekter omhandlende kode udvikling og implementering af forskellige udgaver af terninge- og monopoly spil, vil der i denne rapport takes fat i at optimere disse tidligere udgaver, hvilket leder op til denne sidste udgave af Matador spillet. For at beskrive systemets krav er der først bliver udarbejdet kravanalyse, dernæst er udvalgte af disse krav præsenteret i form af forskellige use cases og use case diagram samt et domain model, hvilket har til formål at visualiserer den grundlæggende struktur for spillet.

Ydermere bliver er i denne rapport sat lyst på systemets funktionalitet og design, ved at udarbejde klassediagramer over hele det udviklede system og sekvensdiagramer over centrale dele af det udviklede system, for at definere software objekter og disses samarbejde og ansvar (Larman, 2004).

For at udvikle systemet er der kodet i IntelliJ i Java UTF-8, samt er der gjort brug af github for at være i stand til at kunne samarbejde fra forskellige computere og kunne versionsstyre projektet.

Derudover vil der i forbindelse med implementering af systemet udvælges kodedele som vil forklares samt dokumenteres for test og endelig vurdering af systemets status kvalitetsmæssigt.

2 KRAV

I det følgende afsnit er der ud fra kundens vision om et Matadorspil der kan spilles på DTU's databarer, med kode af høj kvalitet, dannet en liste krav for produktet. Disse krav vil specificere, hvad der skal implementeres, systemets egenskaber samt hvad systemet kan tilbyde brugerne, hvorfor de er klassificeret ved hjælp af FURPS-modellen.

Functionality

- Spillet skal kunne spilles mellem 2 - 6 personer
- Spillet indeholder 2 terning, hvor terningerne består af 6 sider, der har værdierne mellem 1 - 6
- Spillerne repræsenteres i spillet hver som en bil i selvvalgt farve.
- Spillerne starter fra feltet start
- Spillerne rykker de antal øjne terningerne viser.
- Spillerens bil kan kun rykke fremad og rykkes i ring.
- Hver gang en spiller passerer start, modtager denne spiller 4000 kr. fra banken til sin pengebeholdning.
- Hver spiller starter med en pengebeholdning på 30.000 kr.
- Spillet består af en spilleplade med 40 firkantede felter, hvor hvert felt har sin egen funktion, feltet "start" indgår i de 40 felter.
- Når spilleren lander på et felt, udskrives der en specifik tekst som repræsenterer hvad der kommer til at ske for spilleren på det aktuelle felt.
- Lander en spiller på feltet "Prøv lykken", får spilleren en tilfældig tekst frem på skærmen som repræsentere hvad spilleren skal gøre.
- Hvis en spiller lander på et felt der repræsenterer en grund eller virksomhed som ikke er ejet af nogen, kan spilleren vælge at købe feltet med pengene fra sin pengebeholdning og

dermed vil den spiller eje denne grund, eller kan spilleren vælge ikke at købe feltet, og så sker der intet.

- Hvis en spiller lander på et felt der allerede ejes af en anden spiller, skal man betale husleje fra sin pengebeholdning til den spiller der ejer grunden.
- Hvis en spiller lander på feltet “De fængsles” skal man direkte i fængsel og man modtager ikke de 4000 kr. man får ved at passere start.
- Hvis en spiller lander på feltet “I Fængsel”, er man på besøg og der sker ikke noget.
- Hvis en spiller lander på feltet “Parkering” er det et fristed, og man skal ikke foretage sig noget indtil det bliver ens egen tur igen.
- Hvis en spiller lander på feltet indkomstskat skal spilleren af med 4000 kr. Fra sin pengebeholdning til banken.
- Når man fængsles, sendes man i fængsel og man modtager en bøde på 1000 kr., som man skal betale til banken.
- Spilleren der først ikke kan betale leje til en spiller eller ikke kan betale skat, taber spillet og spillet stopper. Er man flere end to spillere, vil den spiller med størst pengebeholdning vinde spillet.

Usability

- Spillet kan spilles uden først at læse en brugervejledning.
- Spillets sprog er dansk

Performance

- Når terningerne er blevet slået i spillet, må det ikke tage mere end 333 millisekunder før værdien af terningen vises.

Supportability

- Spillet programmeres i Java, dermed skal Java være installeret på computeren for at spillet kan køre.

3 ANALYSE

For at prioritere en høj kvalitet af kode og implementering af få spilleregler, for ikke at gå på kompromis med brugervenligheden, er der blevet arbejdet med use cases, use case diagrammer og domain model. Disse værktøjer har bidraget til et bedre overblik over systemets funktionalitet og prioritering af features. Ved hjælp af Unified Processes er der arbejdet agilt og iterativt, dermed er der under forløbet, i de forskellige iterationer, tilrettet i Use Cases, hvilket har bidraget til at arbejde effektivt og minimere risikoen for eventuelle misforståelser. I det følgende vil de endelige use cases og den endelige domain model blive repræsenteret.

3.1 USE CASES

- **Setup Spil**

Først vælges antal spillere, så indtastes der på tur spillerens navn og valg af farve på bil. Derefter sættes spillernes biler på spillepladen, hvor der også fremgår hver spiller samt deres pengebeholdning på 30.000kr til at starte med.

- **Kast terningerne**

Spillerne skal trykke på knappen OK for at kunne slå med terningerne, derefter rykker spillerne de antal øjne der vises på terningerne.

- **Køb grunde**

Lander en spiller på en ukøbt grund, der dermed ikke er ejet af en anden spiller vil spilleren blive spurgt om de vil købe den pågældende grund, prisen på grunden står på feltet der repræsentere grunden. Vælger spilleren at trykke på ja, vil spilleren eje grunden og betale prisen fra sin pengebeholdning. Vælger spilleren er trykke på nej sker der ingenting.

- **Prøv lykken kort**

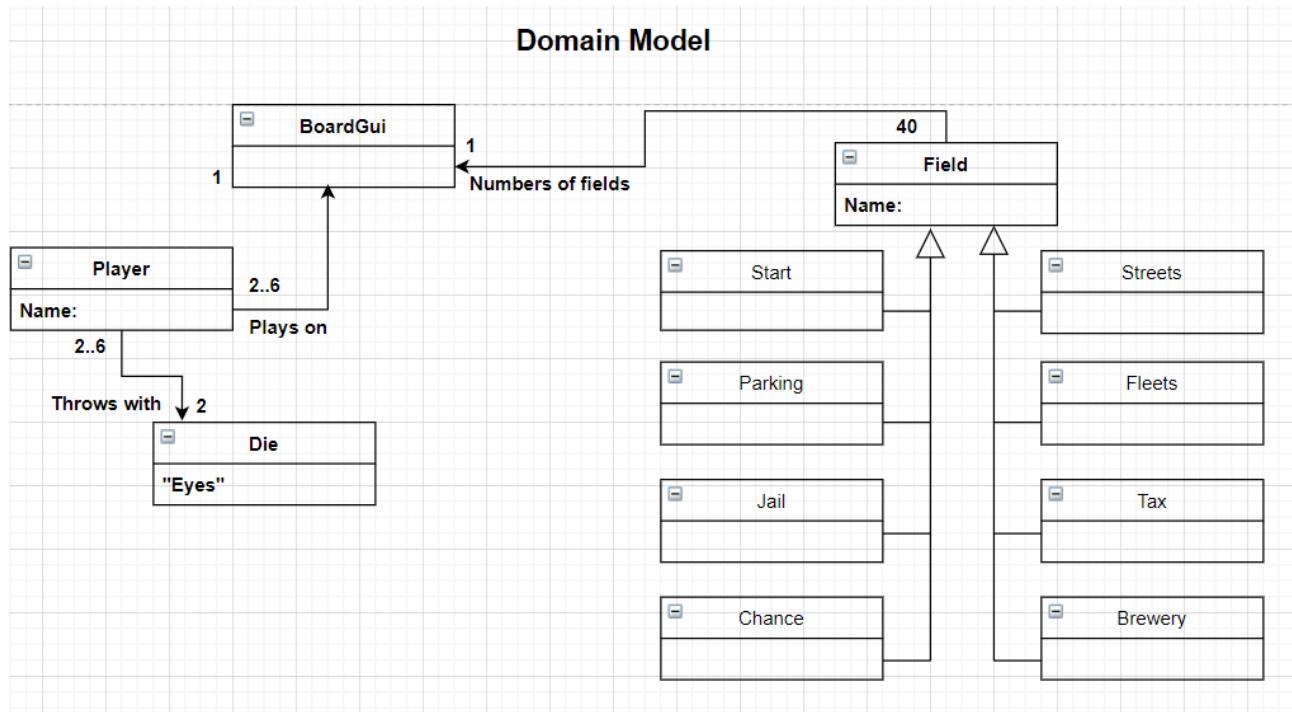
Lander en spiller på et prøv lykken felt, får spilleren et random prøv lykken kort, hvor der er skrevet en tekst, der påvirker spillerens pengebeholdning på negativ eller positiv måde.

UseCase: SpilSpillet
ID: 4
Kort beskrivelse: To til seks spillere starter med hver deres pengebeholdning på 30.000 kr., hvor de på skift kaster med to terninger, værdien af disse terninger, viser hvor mange felter spilleren skal række sin bil i spillet. Feltet spilleren lander på kan vælges at købes af spilleren. Lander en spiller på en grund der er ejet af en anden spiller skal der betales husleje. Den spiller der først ikke kan betale husleje, taber spillet. Den spiller med den største pengebeholdning, når en anden spiller ikke længere kan betale, har vundet spillet.
Primære aktører: Spiller1, spiller2, spiller3, spiller4, spiller5, spiller6
Sekundære aktører: Ingen
Forudsætninger: Computeren skal have en compiler for at spillet kan køre, samt installeret java.
Main flow: <ol style="list-style-type: none">1. Use casen begynder når spiller1 trykker på knappen OK, ved siden af knappen "Slå!" for at slå med terningerne.2. Terningerne viser to random tal med en værdi mellem 2-12. Værdien er det antal felter spillerens bil skal række sig.3. Hvis feltet ikke er ejet af en anden spiller får spilleren muligheden for at købe feltet.4. Ellers<ol style="list-style-type: none">4.1. Skal spilleren betale husleje, til den spiller der ejer feltet.5. Alle felternes priser og egenskaber fremgår på spillepladen, når en spiller landet på feltet.6. Felterne og deres egenskaber har enten en negativ eller positiv påvirkning på spillernes pengebeholdning.7. Den spiller der ikke kan betale en anden spiller har tabt.8. Spilleren der har flest penge i sin pengebeholdning når taberen er fundet er vinderen og spillet afsluttes.
Postconditions: Der er en vinder og taber af spillet.

3.2 DOMAIN MODEL

En Domain model er en måde at repræsentere virkelige objekter og konceptuelle klasser samt forholdet mellem disse, visuelt. Denne visuelle repræsentation, vil medføre en forståelse af systemet og hjælpe udviklere med at designe systemet til vedligeholdelse og trinvis udvikling. Ydermere vil domain modellen også identificere eventuelle misforståelser mellem kunde og projektgruppe, da modellen forestiller systemets funktioner.

Nedenfor ses de forskellige klasser i Matador spillet, samt forholdet mellem dem.

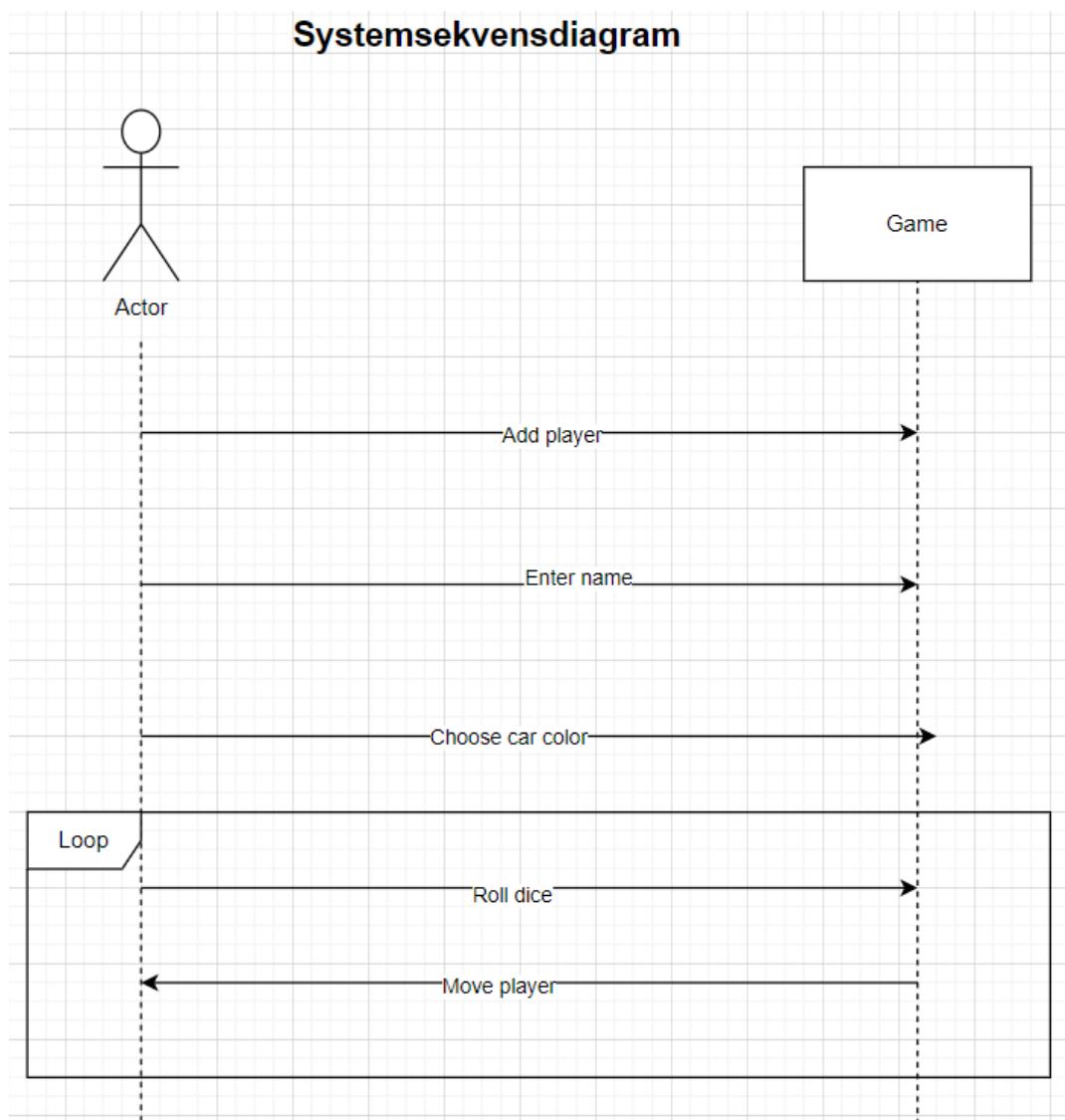


Figur 1 - Domain Model

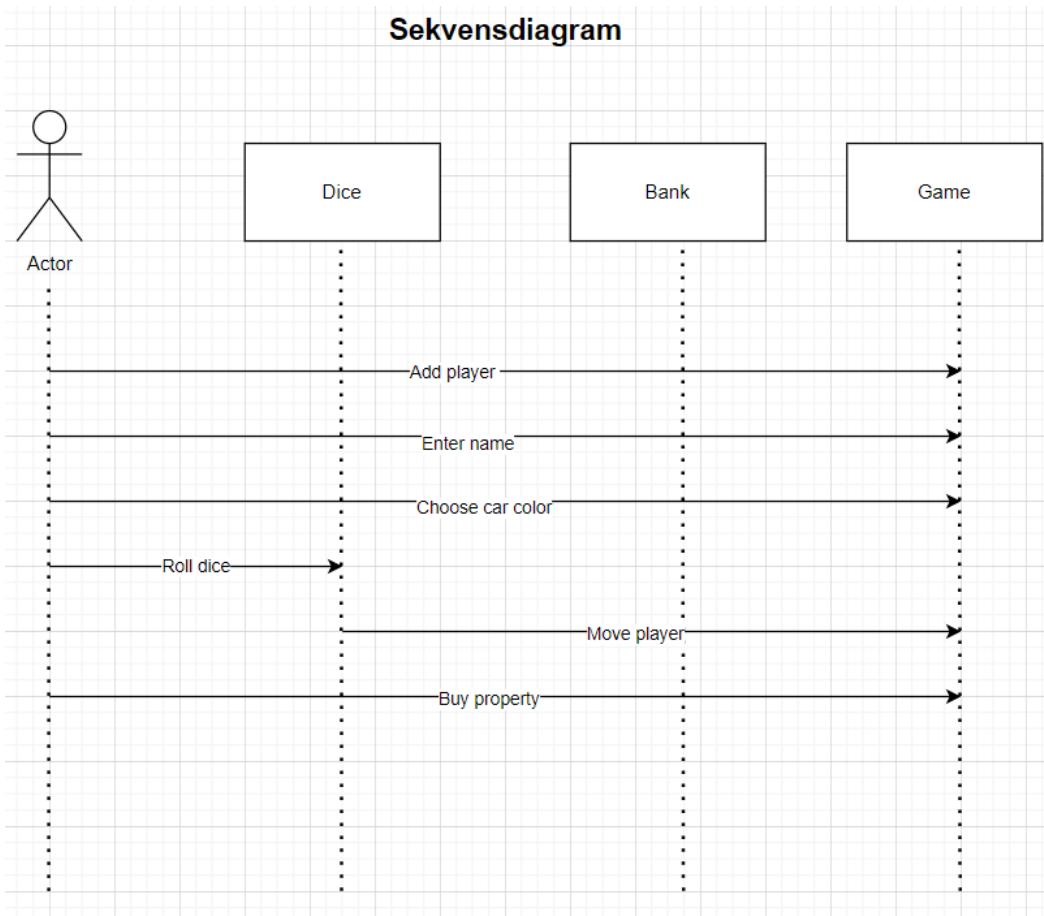
4 DESIGN

Jævnføre den objekt-orienterede process er den konceptuelle domain model og use cases benyttet i den objekt-orienterede design fase til at definere, identificere og designe systemklasser og objekter samt deres forhold. Herunder er der udarbejdet et systemsekvensdiagram, sekvensdiagram og designklassediagram (Larman, 2004).

4.1 SYSTEMSEKVENSDIAGRAM



4.2 SEKVENSDIAGRAM



4.3 DESIGNKLASSEDIAGRAM

Vi henviser til diagrammet i projektmappen, da det er for stort til at tage med i rapporten. Ud fra designklassediagrammet kan det ses hvordan klasserne integrerer med hinanden derudover viser den også hvordan en software klasse spiller en rolle.

5 IMPLEMENTERING

High Cohesion:

High Cohesion hører oftest sammen med Low Coupling. Man opnår High Cohesion ved at sørge for at have fokus på de vigtigste metoder og attributter for ens klasser, så man ikke ender med at skrive en masse overflødig kode.

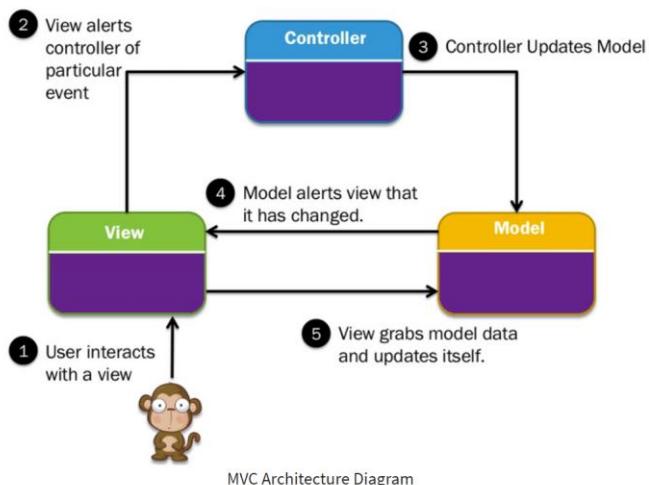
Low Coupling:

Low Coupling opnås ved at sørge for programmets klasser er så uafhængige som muligt. Dette gør at man nemt kan genbruge klasser eller lave ændringer i en klasse uden det forårsager direkte problemer.

MVC Framework:

Vi har genbrugt terningerne fra vores CDIO_2, da de uafhængige af andre klasse pga. er Low Coupling. Vi har fulgt MVC framework under implementeringen af spillet. MVC/Model-View-Controller framework er en model som gør det nemmere at dele ansvaret ud til de forskellige komponenter. Al data og logik der tilhører dataene, hører under Model. Controller er mellemmanden til View og Model, hvor View er den der præsenterer dataene til brugeren fx ved brug af en GUI.

MVC Architecture



1

Vores GameController hører under Controllers. Player, Die, ChanceCard, Field og alle tilhørende subklasser hører til Models. Til sidst har vi Views som er vores BoardGui klasse.

Vi har altså valgt at lave spil logikken i vores GameController. Vi har lavet de konstruktører og get/set metoder vi vurderede, var nødvendige. I dette afsnit tages der fat på nogle bestemte dele af projektet, nedarvning, casting af fields, metoden payRentFleet().

¹<https://www.guru99.com/mvc-tutorial.html>

Koden

Vi har gjort brug af nedarvning af vores felter. Klassen Field er en abstractklasse, som kun har attributterne name, bgColor og fgColor, med de passende get og set metoder til. Klasserne Street, Start, Chance, Parking, Jail, JailVisit, Tax, Fleet og Brewery er alle nedarvet fra Field.

Vi har lavet et array af fields i BoardGUI klassen, hvor vi opretter de fields der repræsentere de felter som skal vises på GUI'en. Nedenfor ses de forskellige instancer af fields, i arrayet.

```
8 public class BoardGUI {
9     public static Field[] fields = {
10         new Start( name: "START", Color.red, Color.BLACK, Profit: 4000, subText: "Modtag kr. 4000"),
11         new Street( name: "Rødvarevej", subText: "1.200KR", price: 1200, Color.blue, Color.BLACK, rent1: 1200),
12         new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
13         new Street( name: "Hvidovrevej", subText: "1.200KR", price: 1200, Color.blue, Color.BLACK, rent1: 1200),
14         new Tax( name: "Skat", Color.cyan, Color.BLACK, taxFee: 4000, subText: "Betal 4000KR"),
15         new Fleet( name: "Scandilines", subText: "Færge", Price: 4000, Color.BLUE, Color.BLACK),
16         new Street( name: "Rodskildevej", subText: "2.000KR", price: 2000, Color.ORANGE, Color.BLACK, rent1: 2000),
17         new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
18         new Street( name: "Valby Langgade", subText: "2.000KR", price: 2000, Color.ORANGE, Color.BLACK, rent1: 2000),
19         new Street( name: "Allégade", subText: "2.400KR", price: 2400, Color.ORANGE, Color.BLACK, rent1: 2400),
20         new JailVisit( name: "Fængsel", subText: "På besøg", Color.gray, Color.BLACK),
21         new Street( name: "Frederiksberg Allé", subText: "2.800KR", price: 2800, Color.GREEN, Color.BLACK, rent1: 2800),
22         new Brewery( name: "Turborg Squash", subText: "3.000KR", price: 3000, rent: 100, Color.CYAN, Color.BLACK),
23         new Street( name: "Gülowvej", subText: "2.800KR", price: 2800, Color.GREEN, Color.BLACK, rent1: 2800),
24         new Street( name: "Gl.Kongevej", subText: "3.200KR", price: 3200, Color.GREEN, Color.BLACK, rent1: 3200),
25         new Fleet( name: "Mols-Linien", subText: "4.000kr", Price: 4000, Color.RED, Color.BLACK),
26         new Street( name: "Bernstorfftsvej", subText: "3.600KR", price: 3600, Color.GRAY, Color.BLACK, rent1: 3600),
27         new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
28         new Street( name: "Hellerupvej", subText: "3.600KR", price: 3600, Color.GRAY, Color.BLACK, rent1: 3600),
29         new Street( name: "Strandvejen", subText: "4000KR", price: 4000, Color.GRAY, Color.BLACK, rent1: 4000),
30         new Parking( name: "Parking", subText: "Parking", Color.BLUE, Color.BLACK),
31         new Street( name: "Trianglen", subText: "4.400KR", price: 4400, Color.RED, Color.BLACK, rent1: 3500),
32         new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
```

For at felterne bliver vist på GUI'en har vi lavet en metode guiFieldConvert(), som tager et Field[] som parameter og caster vores felter og laver dem om til guiFields. Som ses nedenfor har vi brugt felternes get-metoder.

```
6 public class ConvertHelper {
7     @
8         public static GUI_Field[] guiFieldConvert(Field[] fields) {
9             GUI_Field[] gui_fields = new GUI_Field[fields.length];
10            for (int i = 0; i < fields.length; i++) {
11                if (fields[i] instanceof Street) {
12                    gui_fields[i] = new GUI_Street();
13                    gui_fields[i].setTitle(fields[i].getName());
14                    gui_fields[i].setSubText(((Street) fields[i]).getSubText());
15                    gui_fields[i].setBackGroundColor(fields[i].getBgColor());
16                    gui_fields[i].setForeGroundColor(fields[i].getFgColor());
17                } else if (fields[i] instanceof Brewery) {
18                    gui_fields[i] = new GUI_Brewery();
19                    gui_fields[i].setTitle(fields[i].getName());
```

Vi har lavet en landOnField metode, dog er denne metode lavet i klassen GameController, vi har derfor ikke brugt polymorphi ift. denne metode. En anden måde vi kunne have gjort dette, med mere fokus på polymorphi er hvis man lavede en landOnField metode i Field klassen. Denne metode ville

så blive arvet af sub-klasserne. Man kunne derfor override landOnField metoden og tilføje den unikke logik der skal være for hvert felt.

```
private void landOnField(Field field, Player player) {
    if (field instanceof Street) {
        if (!((Street) field).getOwner()) {
            if (gui.getUserLeftButtonPressed( msg: "Vil du købe denne grund?", trueButton: "Ja", falseButton: "Nej")) {
                ((Street) field).setHasOwner(true);
                player.gui_player.setBalance(player.gui_player.getBalance() - ((Street) field).getPrice());
                owner = player;
                // Caster til ownable og sætter border til ejers farve
                GUI_Ownable ownable = (GUI_Ownable) gui.getFields()[owner.playerPosition];
                ownable.setBorder(owner.gui_player.getPrimaryColor());
            }
        }
    }
}
```

Vi har altså her lavet den i GameController klassen, med en række if-sætninger som tjekker hvilket slags felt man lander på. Inde i metoden kalder vi metoderne payRentStreet(), payRentFleet(), payRentBrewery(), hvor payRentStreet() og payRentBrewery() kun sørger for at overføre penge så holder payRentFleet() også øje med hvor mange Fleets spilleren ejer, da lejen ændre sig jo flere man har.

```
//holder øje med hvor mange færger spilleren har så lejen ændre sig
private void payRentFleet(Field field, Player player) {
    if (field instanceof Fleet) {
        switch (owner.getOwnedFleets()) {
            case 1:
                if (player.gui_player.getBalance() >= ((Fleet) field).getFleet1()) {
                    gui.showMessage( msg: "Du skal betale " + ((Fleet) field).getFleet1() + "KR, til " + owner.gui_player.getName());
                    player.gui_player.setBalance(player.gui_player.getBalance() - ((Fleet) field).getFleet1());
                    owner.gui_player.setBalance(owner.gui_player.getBalance() + (((Fleet) field).getFleet1()));
                } else {
                    gui.showMessage( msg: "Du har ikke nok penge til at betale ejeren.");
                }
                break;
            case 2:
                if (player.gui_player.getBalance() >= ((Fleet) field).getFleet2()) {
                    gui.showMessage( msg: "Du skal betale " + ((Fleet) field).getFleet2() + "KR, til " + owner.gui_player.getName());
                    player.gui_player.setBalance(player.gui_player.getBalance() - ((Fleet) field).getFleet2());
                    owner.gui_player.setBalance(owner.gui_player.getBalance() + (((Fleet) field).getFleet2()));
                } else {
                    gui.showMessage( msg: "Du har ikke nok penge til at betale ejeren.");
                }
                break;
            case 3:
        }
    }
}
```

Vi har lavet en switch case som tjekker hvor mange ownedFleets spilleren har. ownedFleets variablen kommer fra Playerklassen.

6 TEST

Vi bedømte det ikke nødvendigt at lave JUnit tests for at kunne udføre de korrekte tests, vi har dog valgt at lave en manuel test for at søge efter fejl. Vi brugte en terning og satte den til kun at kaste værdien 1, så spillerne rykker sig et felt ad gangen. Man kan købe grunde, færger og bryggerier uden problemer. Ejers bil farve bruges til at vise ejeren af grunden.

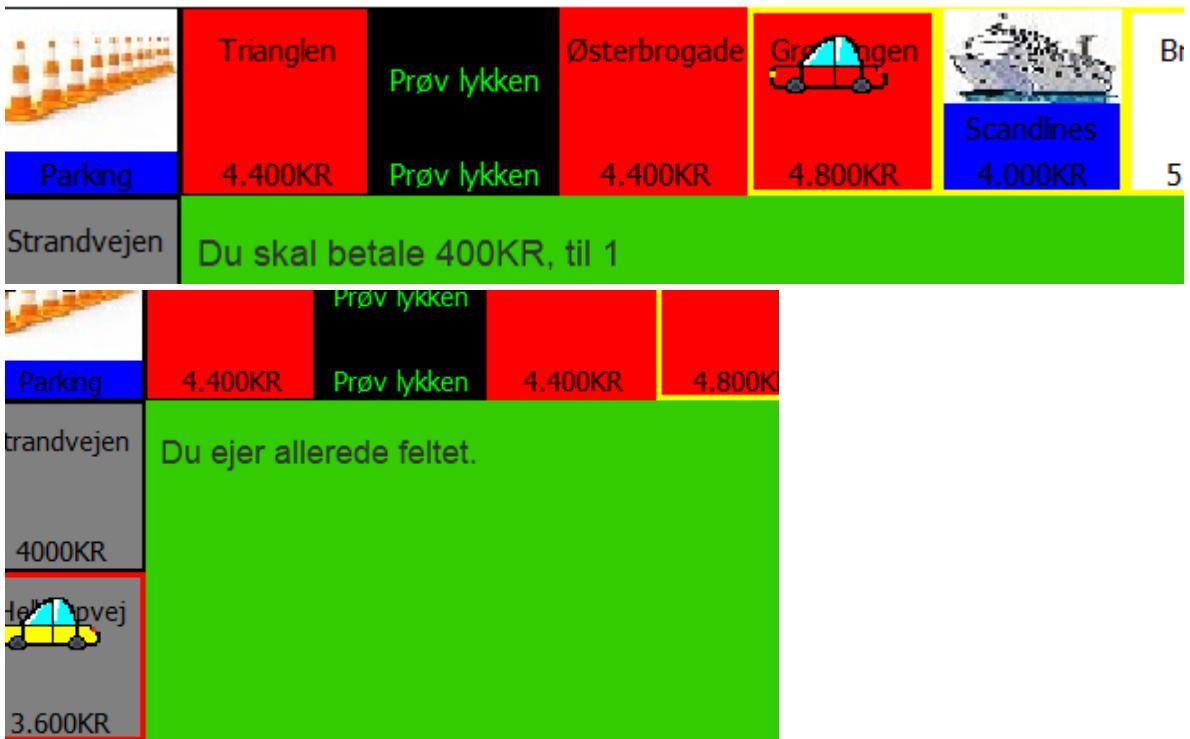
			2 20900		ordinær Statskat
			1 22200		Betal 2000KR
lykken			Skat	Hvidovrevej	Rådhuspladse
lykken	2.000KR	Scandines Færge	Betal 4000KR	1.200KR	8.000KR
				Prøv lykken	Prøv lykken
				Rødovrevej	START
					subText

	Trianglen		Prøv lykken		Grønningen		Bredgade	Kgs.Nytorv	Østergade	
Parking	4.400KR	Prøv lykken	4.400KR	4.800KR	Scandines	4.000KR	5.200KR	5.200KR	Coca cola	3.000KR
Strandvejen	Vil du købe denne grund?									Østergade
4000KR										6.000KR

Vi fandt dog en fejl som nogle gange forekommer. Fejlen forekommer kun på Street felter og sker kun nogle gange. Som ses forneden bliver ejeren (den gule bil) bedt om at betale penge til den røde (spiller 2).

	Trianglen		Prøv lykken		Østerborgade	Grønningen		Bredgade	Kgs.Nytorv	Østergade	
Parking	4.400KR	Prøv lykken	4.400KR	4.800KR	Scandines	4.000KR	5.200KR	5.200KR	Coca cola	3.000KR	5.600KR
Strandvejen	Du skal betale 450KR, til 2									Amagertorv	
4000KR											6.000KR

Vi kørte spillet et par gange efter at have prøvet på at rette fejlen, det virkede dog ikke. Spillet kan nogle gange forveksle ejeren. For neden kan det ses at programmet vælger samme ejer for alle street felterne. Vi har desværre ikke kunne finde ud af hvor og hvorfor fejlen opstår præcis, da dette kun sker for Street felterne.



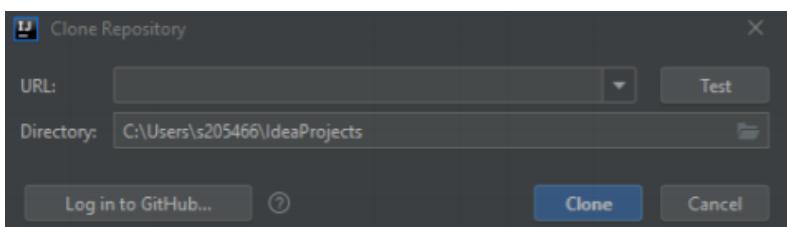
Vi har udførte også en brugertest, med brugere som ikke kender til kode. Vi udførte brugertesten på en gruppe af 4 drenge. De kunne nemt starte spillet, de vidste hvad de skulle trykke på og hvordan de skulle vælge farver. De blev dog nemt forvirrede over om de selv skulle trykke på en knap for at betale penge til ejeren af en grund eller om det skete automatisk. Vi rettede dette ved at skrive flere informerende tekster hver gang der sker noget nyt for at undgå disse forvirringer. Det gik også hurtigt op for dem at der var en fejl som ville opstå nogle gange og at de endte med at betale leje for deres egne grunde.

7 KONFIGURATION STYRING

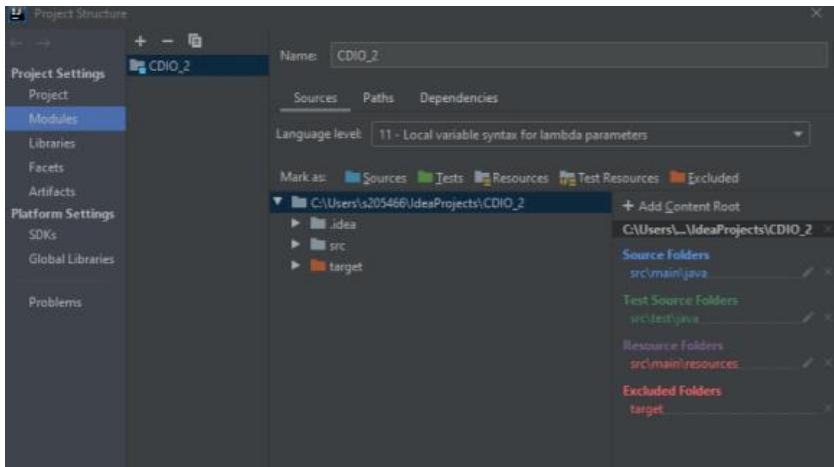
“For at kunne køre spillet og dets tests, skal IntelliJ (version 2019.2.1 og op) være downloadet.

Koden hentes fra vores Git repository, linket til Git repository ligger for enden af afsnittet.

Åben IntelliJ og tryk på ”File” → ”New” → ”Project from version control” → ”Git”, derefter insæt linket til Git repository der hvor der står URL.



Når dette er gjort, skal man sørge for at man kører de rigtige indstillinger, dette gør man ved at trykke på ”Settings” under ”File”, og folde ”Build, Execution, Deployment” → folde ”compiler” ud også trykke ”java compiler”. Under ”target bytecode version” skal der ikke stå noget, det er derfor vigtigt at tjekke om der står noget. Dernæst skal man åbne ”open module settings”, den finder man ved at højre klikke på projektetmappen. Man skal så sørge for at ”language level” er sat til at være 11.



Nu kan man køre spillet. Der er mulighed for at Maven ikke er blevet importeret korrekt når man først importerer projektet, hvis dette sker og man ikke kan køre programmet, så skal man højreklikke på filen ”pom.xml” → klikke ”maven” → ”Reimport”. Når alt er reimporteret, så er problemet løst. Der er to pom.xml filer, det kan være nødtvendigt at gøre dette på begge. Der er dog samtidigt også mulighed for at dette trin ikke er nødvendigt.”²

8 MÄGLER

Vi havde problemer med at implementere køb af huse og hoteller. For at følge spillet regler skal man eje alle grunde i samme farve for at kunne bygget huse, spillet ville dog stoppe med at køre så snart den kode blev implementeret. Grundet tidspres kunne vi ikke få fikset dette i tide, vi valgte derfor at fjerne det for at sørge for resten fungerede. Vi har derudover lavet chancekort, men meget få. Det har ikke været en prioritet at lave alle chancekortene i spillet, da vi vurderede dette til ikke at være nødvendigt for spillet men ”Nice to have”.

² 18_del3, vores rapport.

9 KONKLUSION

Gruppen har formået at fremstille et fungerende matadorspil, med nogle få manglende features. Vi har formået at udnytte viden vi har lært i løbet af 13-ugers kurset, og opfyldt kravene. Spillet har nogle få fejl og kunne optimeres med mere tid. Dog grundet tidspres har vi ikke kunne nå alle vores mål.

10 BILAG

```
1 package com.company.Controllers.helper;
2
3 import com.company.Models.Fields.*;
4 import gui_fields.*;
5
6 public class ConvertHelper {
7     @
8         public static GUI_Field[] guiFieldConvert(Field[] fields) {
9             GUI_Field[] gui_fields = new GUI_Field[fields.length];
10            for (int i = 0; i < fields.length; i++) {
11                if (fields[i] instanceof Street) {
12                    gui_fields[i] = new GUI_Street();
13                    gui_fields[i].setTitle(fields[i].getName());
14                    gui_fields[i].setSubText(((Street) fields[i]).getSubText());
15                    gui_fields[i].setBackGroundColor(fields[i].getBgColor());
16                    gui_fields[i].setForeGroundColor(fields[i].getFgColor());
17                } else if (fields[i] instanceof Brewery) {
18                    gui_fields[i] = new GUI_Brewery();
19                    gui_fields[i].setTitle(fields[i].getName());
20                    gui_fields[i].setSubText(((Brewery) fields[i]).getSubText());
21                    gui_fields[i].setBackGroundColor(fields[i].getBgColor());
22                    gui_fields[i].setForeGroundColor(fields[i].getFgColor());
23                } else if (fields[i] instanceof Chance) {
24                    gui_fields[i] = new GUI_Chance();
25                    gui_fields[i].setTitle(fields[i].getName());
26                    gui_fields[i].setBackGroundColor(fields[i].getBgColor());
27                } else if (fields[i] instanceof Fleet) {
28                    gui_fields[i] = new GUI_Shipping();
```

```
29             gui_fields[i].setTitle(fields[i].getName());
30             gui_fields[i].setSubText(((Fleet) fields[i]).getSubText());
31             gui_fields[i].setBackGroundColor(fields[i].getBgColor());
32             gui_fields[i].setForeGroundColor(fields[i].getFgColor());
33         } else if (fields[i] instanceof Tax) {
34             gui_fields[i] = new GUI_Tax();
35             gui_fields[i].setTitle(((Tax) fields[i]).getName());
36             gui_fields[i].setSubText(((Tax) fields[i]).getSubText());
37             gui_fields[i].setBackGroundColor(fields[i].getBgColor());
38             gui_fields[i].setForeGroundColor(fields[i].getFgColor());
39         } else if (fields[i] instanceof Parking) {
40             gui_fields[i] = new GUI_Refuge();
41             gui_fields[i].setTitle(((Parking) fields[i]).getName());
42             gui_fields[i].setSubText(((Parking) fields[i]).getSubText());
43             gui_fields[i].setBackGroundColor(fields[i].getBgColor());
44             gui_fields[i].setForeGroundColor(fields[i].getFgColor());
45         } else if (fields[i] instanceof Start) {
46             gui_fields[i] = new GUI_Start();
47             gui_fields[i].setTitle(fields[i].getName());
48             gui_fields[i].setBackGroundColor(fields[i].getBgColor());
49             gui_fields[i].setForeGroundColor(fields[i].getFgColor());
50         } else if (fields[i] instanceof Jail) {
51             gui_fields[i] = new GUI_Jail();
52             gui_fields[i].setTitle(fields[i].getName());
53             gui_fields[i].setSubText(((Jail) fields[i]).getSubText());
54             gui_fields[i].setBackGroundColor(fields[i].getBgColor());
55             gui_fields[i].setForeGroundColor(fields[i].getFgColor());
56     } else if (fields[i] instanceof JailVisit) {
```

```
57             gui_fields[i] = new GUI_Jail();
58             gui_fields[i].setTitle(fields[i].getName());
59             gui_fields[i].setSubText(((JailVisit) fields[i]).getSubText());
60             gui_fields[i].setBackGroundColor(fields[i].getBgColor());
61             gui_fields[i].setForeGroundColor(fields[i].getFgColor());
62         }
63     }
64     return gui_fields;
65 }
66 }
```

```
1 package com.company.Controllers;
2
3 import com.company.Models.ChanceCard;
4 import com.company.Models.Die;
5 import com.company.Models.Fields.*;
6 import com.company.Models.Player;
7 import com.company.Views.BoardGUI;
8 import gui_fields.GUI_Ownable;
9 import gui_main.GUI;
10
11 import java.awt.*;
12
13 public class GameController {
14
15     public Player[] player;
16     public GUI gui;
17     private int turn = 1;
18     Die die1 = new Die( MIN: 1, MAX: 6);
19     Die die2 = new Die( MIN: 1, MAX: 6);
20     boolean.isPlaying = true;
21     public Player owner;
22
23     public ChanceCard chanceCard = new ChanceCard();
24
25     public GameController(GUI gui) { this.gui = gui; }
26
27     private void setUpPlayers() {
```

```

30
31     String maxPlayers = gui.getUserSelection( msg: "Vælg antal spillere", ...options: "2", "3", "4", "5", "6");
32     player = new Player[Integer.parseInt(maxPlayers)];
33     for (int i = 0; i < player.length; i++) {
34         String colors = gui.getUserSelection( msg: "Vælg en farve.", ...options: "Gul", "Rød", "Grøn", "Blå", "Hvid", "Sort");
35         Color color;
36
37         switch (colors) {
38             case "Gul":
39                 color = Color.YELLOW;
40                 break;
41             case "Rød":
42                 color = Color.RED;
43                 break;
44             case "Grøn":
45                 color = Color.GREEN;
46                 break;
47             case "Blå":
48                 color = Color.BLUE;
49                 break;
50             case "Hvid":
51                 color = Color.WHITE;
52                 break;
53             case "Sort":
54                 color = Color.BLACK;
55                 break;

```

```

57                     color = Color.CYAN;
58     }
59     player[i] = new Player(gui, color);
60     gui.addPlayer(player[i].getGui_player());
61     gui.getFields()[0].setCar(player[i].getGui_player(), hasCar: true);
62 }
63
64
65     private void calculatePlayerTurn() {
66         if (player.length == 2) {
67             while (true) {
68                 switch (turn) {
69                     case 1:
70                         movePlayer(player[0]);
71                         checkBalance();
72                         turn = 2;
73                         break;
74                     case 2:
75                         turn = 1;
76                         movePlayer(player[1]);
77                         checkBalance();
78                         break;
79
80             }
81         }
82     }

```

```
83     if (player.length == 3) {
84         while (true) {
85             switch (turn) {
86                 case 1:
87                     movePlayer(player[0]);
88                     checkBalance();
89                     turn = 2;
90                     break;
91                 case 2:
92                     movePlayer(player[1]);
93                     checkBalance();
94                     turn = 3;
95                     break;
96                 case 3:
97                     movePlayer(player[2]);
98                     checkBalance();
99                     turn = 1;
100    }
101    }
102    }
103    if (player.length == 4) {
104        while (true) {
105            switch (turn) {
106                case 1:
107                    turn = 2;
108                    movePlayer(player[0]);
109                    checkBalance();
```

```
110                     break;
111
112             case 2:
113                 turn = 3;
114                 movePlayer(player[1]);
115                 checkBalance();
116                 break;
117             case 3:
118                 turn = 4;
119                 movePlayer(player[2]);
120                 checkBalance();
121                 break;
122             case 4:
123                 turn = 1;
124                 movePlayer(player[3]);
125                 checkBalance();
126                 break;
127         }
128     }
129     if (player.length == 5) {
130         while (true) {
131             switch (turn) {
132                 case 1:
133                     turn = 2;
134                     movePlayer(player[0]);
135                     checkBalance();
136                     break;
```

```
137         case 2:
138             turn = 3;
139             movePlayer(player[1]);
140             checkBalance();
141             break;
142         case 3:
143             turn = 4;
144             movePlayer(player[2]);
145             checkBalance();
146             break;
147         case 4:
148             turn = 5;
149             movePlayer(player[3]);
150             checkBalance();
151             break;
152         case 5:
153             turn = 1;
154             movePlayer(player[4]);
155             checkBalance();
156             break;
157     }
158 }
159 }
160 if (player.length == 6) {
161     while (true) {
162         switch (turn) {
163             case 1:
```

```

164                     turn = 2;
165                     movePlayer(player[0]);
166                     checkBalance();
167                     break;
168                 case 2:
169                     turn = 3;
170                     movePlayer(player[1]);
171                     checkBalance();
172                     break;
173                 case 3:
174                     turn = 4;
175                     movePlayer(player[2]);
176                     checkBalance();
177                     break;
178                 case 4:
179                     turn = 5;
180                     movePlayer(player[3]);
181                     checkBalance();
182                     break;
183                 case 5:
184                     turn = 6;
185                     movePlayer(player[4]);
186                     checkBalance();
187                     break;
188                 case 6:
189                     turn = 1;
190                     movePlayer(player[5]);
191                     checkBalance();
192                     break;
193             }
194         }
195     }
196 }
197
198 @
199 private void movePlayer(Player player) {
200     gui.showMessage( msg: player.getName() + "'s tur.");
201     String playButton = gui.getUserSelection( msg: "Tryk på OK for at slå med terningerne", ...options: "Slå!");
202
203     if (playButton.equals("Slå!")) {
204         int sum = die1.diceTurn(die1) + die2.diceTurn(die2);
205         gui.setDice(die1.diceNumber, die1.diceNumber);
206
207         gui.getFields()[player.playerPosition].setCar(player.gui_player, hasCar: false);
208         player.playerPosition += sum;
209         if (player.playerPosition >= BoardGUI.fields.length) {
210             player.playerPosition -= BoardGUI.fields.length;
211             gui.showMessage( msg: "Tillykke du modtager 4000Kr for at passere START!!!");
212             player.gui_player.setBalance(player.gui_player.getBalance() + 4000);
213         }
214         gui.getFields()[player.playerPosition].setCar(player.gui_player, hasCar: true);
215         landOnField(BoardGUI.fields[player.playerPosition], player);
216     }
217 }
```

```

218     private void landOnField(Field field, Player player) {
219         if (field instanceof Street) {
220             if (!((Street) field).getOwner()) {
221                 if (gui.getUserLeftButtonPressed( msg: "Vil du købe denne grund?", trueButton: "Ja", falseButton: "Nej")) {
222                     ((Street) field).setHasOwner(true);
223                     player.gui_player.setBalance(player.gui_player.getBalance() - ((Street) field).getPrice());
224                     owner = player;
225                     // Caster til ownable og sætter border til ejers farve
226                     GUI_Ownable ownable = (GUI_Ownable) gui.getFields()[owner.playerPosition];
227                     ownable.setBorder(owner.gui_player.getPrimaryColor());
228                 } else {
229                     ((Street) field).setHasOwner(false);
230                 }
231             } else {
232                 if (owner != player) {
233                     gui.showMessage( msg: "Feltet er allerede ejet.");
234                     payRentStreet(BoardGUI.fields[player.playerPosition], player);
235                 } else {
236                     gui.showMessage( msg: "Du ejer allerede feltet.");
237                 }
238             }
239         } else if (field instanceof Brewery) {
240             if (!((Brewery) field).isHasOwner()) {
241                 if (gui.getUserLeftButtonPressed( msg: "Vil du købe dette bryggeri?", trueButton: "Ja", falseButton: "Nej")) {
242                     ((Brewery) field).setHasOwner(true);
243                     player.gui_player.setBalance(player.gui_player.getBalance() - ((Brewery) field).getPrice());
244                     owner = player;
245
246                     owner.setOwnedBreweries();
247                     GUI_Ownable ownable = (GUI_Ownable) gui.getFields()[owner.playerPosition];
248                     ownable.setBorder(owner.gui_player.getPrimaryColor());
249                 } else {
250                     ((Brewery) field).setHasOwner(false);
251                 }
252             } else {
253                 if (owner != player) {
254                     gui.showMessage( msg: "Bryggeriet er allerede ejet.");
255                     payRentBrewery(BoardGUI.fields[player.playerPosition], player);
256                 } else {
257                     gui.showMessage( msg: "Du ejer allerede feltet.");
258                 }
259             }
260         } else if (field instanceof Fleet) {
261             if (!((Fleet) field).isHasOwner()) {
262                 if (gui.getUserLeftButtonPressed( msg: "Vil du købe denne farge?", trueButton: "Ja", falseButton: "Nej")) {
263                     ((Fleet) field).setHasOwner(true);
264                     player.gui_player.setBalance(player.gui_player.getBalance() - ((Fleet) field).getPrice());
265                     owner = player;
266                     owner.setOwnedFleets();
267                     GUI_Ownable ownable = (GUI_Ownable) gui.getFields()[owner.playerPosition];
268                     ownable.setBorder(owner.gui_player.getPrimaryColor());
269                 } else {
270                     ((Fleet) field).setHasOwner(false);
271                 }
272             }
273         }
274     }

```

```

271     } else {
272         if (player != owner) {
273             gui.showMessage( msg: "Færgen er allerede ejet.");
274             payRentFleet(BoardGUI.fields[player.playerPosition], player);
275         } else {
276             gui.showMessage( msg: "Du ejer allerede fæltet.");
277         }
278     }
279 } else if (field instanceof Tax) {
280     gui.showMessage( msg: "Du skal betale " + ((Tax) field).getTaxFee() + " Kr i SKAT");
281     player.gui_player.setBalance(player.gui_player.getBalance() - ((Tax) field).getTaxFee());
282 } else if (field instanceof Jail) {
283     gui.showMessage( msg: "Du sendes i fængsel og skal betale en bøde på " + ((Jail) field).getJailFee() + " Kr");
284     player.gui_player.setBalance(player.gui_player.getBalance() - ((Jail) field).getJailFee());
285     gui.getFields()[player.playerPosition].setCar(player.gui_player, hasCar: false);
286     player.playerPosition = 10;
287     gui.getFields()[player.playerPosition].setCar(player.gui_player, hasCar: true);
288 } else if (field instanceof JailVisit) {
289     gui.showMessage( msg: "Du er på besøg i fængslet.");
290 } else if (field instanceof Chance) {
291     switch (chanceCard.randomCard()) {
292         case 1:
293             payCarInsurance(player);
294             break;
295         case 2:
296             lotteryCard(player);
297             break;

```

```

298         case 3:
299             goToJail(player);
300             break;
301         case 4:
302             stockCard(player);
303             break;
304         case 5:
305             parkingTicket(player);
306             break;
307         case 6:
308             carTires(player);
309             break;
310     }
311 }
312 }

313 //Metode som sørger for at spillerne kan betale leje for Street
314 private void payRentStreet(Field field, Player player) {
315     if (field instanceof Street) {
316
317         gui.showMessage( msg: "Du skal betale " + ((Street) field).getRent1() + " KR, til " + owner.gui_player.getName());
318         if (player.gui_player.getBalance() >= ((Street) field).getRent1()) {
319             player.gui_player.setBalance(player.gui_player.getBalance() - ((Street) field).getRent1());
320             owner.gui_player.setBalance(owner.gui_player.getBalance() + ((Street) field).getRent1());
321         } else {
322             gui.showMessage( msg: "Du har ikke nok penge til at betale ejeren.");
323         }
324     }
325 }

```

```

325     }
326 }
327
328 //holder øje med hvor mange færger spilleren har så lejen ændre sig
329 private void payRentFleet(Field field, Player player) {
330     if (field instanceof Fleet) {
331         switch (owner.getOwnedFleets()) {
332             case 1:
333                 if (player.gui_player.getBalance() >= ((Fleet) field).getFleet1()) {
334                     gui.showMessage( msg: "Du skal betale " + ((Fleet) field).getFleet1() + "KR, til " + owner.gui_player.getName());
335                     player.gui_player.setBalance(player.gui_player.getBalance() - ((Fleet) field).getFleet1());
336                     owner.gui_player.setBalance(owner.gui_player.getBalance() + (((Fleet) field).getFleet1()));
337                 } else {
338                     gui.showMessage( msg: "Du har ikke nok penge til at betale ejeren.");
339                 }
340                 break;
341             case 2:
342                 if (player.gui_player.getBalance() >= ((Fleet) field).getFleet2()) {
343                     gui.showMessage( msg: "Du skal betale " + ((Fleet) field).getFleet2() + "KR, til " + owner.gui_player.getName());
344                     player.gui_player.setBalance(player.gui_player.getBalance() - ((Fleet) field).getFleet2());
345                     owner.gui_player.setBalance(owner.gui_player.getBalance() + (((Fleet) field).getFleet2()));
346                 } else {
347                     gui.showMessage( msg: "Du har ikke nok penge til at betale ejeren.");
348                 }
349                 break;
350             case 3:
351                 if (player.gui_player.getBalance() >= ((Fleet) field).getFleet3()) {
352                     gui.showMessage( msg: "Du skal betale " + ((Fleet) field).getFleet3() + "KR, til " + owner.gui_player.getName());
353                     player.gui_player.setBalance(player.gui_player.getBalance() - ((Fleet) field).getFleet3());
354                     owner.gui_player.setBalance(owner.gui_player.getBalance() + (((Fleet) field).getFleet3()));
355                 } else {
356                     gui.showMessage( msg: "Du har ikke nok penge til at betale ejeren.");
357                 }
358                 break;
359             case 4:
360                 if (player.gui_player.getBalance() >= ((Fleet) field).getFleet4()) {
361                     gui.showMessage( msg: "Du skal betale " + ((Fleet) field).getFleet4() + "KR, til " + owner.gui_player.getName());
362                     player.gui_player.setBalance(player.gui_player.getBalance() - ((Fleet) field).getFleet4());
363                     owner.gui_player.setBalance(owner.gui_player.getBalance() + (((Fleet) field).getFleet4()));
364                 } else {
365                     gui.showMessage( msg: "Du har ikke nok penge til at betale ejeren.");
366                 }
367             }
368         }
369     }
370 }
371
372 private void payRentBrewery(Field field, Player player) {
373     if (field instanceof Brewery) {
374         if (player.gui_player.getBalance() >= 100) {
375             gui.showMessage( msg: "Du skal betale " + 100 + "KR, til " + owner.gui_player.getName());
376             player.gui_player.setBalance(player.gui_player.getBalance() - 100);
377             owner.gui_player.setBalance(owner.gui_player.getBalance() + 100);
378         } else {
379             gui.showMessage( msg: "Du har ikke nok penge til at betale ejeren.");
380         }
381     }

```

```

381     }
382
383     //Chancekort
384     @private void goToJail(Player player) {
385         gui.displayChanceCard( txt: "CHANCEKORT: Du er blevet taget for at køre for hurtigt. Du fængsles, og mister 1000KR. Du modtager ikke 4000KR for at få udtaget fra fængslet." );
386         gui.getFields()[player.playerPosition].setCar(player.gui_player, hasCar: false);
387         player.playerPosition = 10;
388         gui.getFields()[player.playerPosition].setCar(player.gui_player, hasCar: true);
389         player.gui_player.setBalance((player.gui_player.getBalance() - 1000));
390         gui.displayChanceCard();
391     }
392
393     @private void payCarInsurance(Player player) {
394         gui.displayChanceCard( txt: "CHANCEKORT: Betal din bilforsikring på " + 1000 + "KR" );
395         player.gui_player.setBalance(player.gui_player.getBalance() - 1000);
396         gui.displayChanceCard();
397     }
398
399     @private void lotteryCard(Player player) {
400         gui.displayChanceCard( txt: "CHANCEKORT: Du har vundet klasselotteriet, du modtager derfor " + 500 + "KR" );
401         player.gui_player.setBalance(player.gui_player.getBalance() + 500);
402         gui.displayChanceCard();
403     }
404
405     @private void stockCard(Player player) {
406         gui.displayChanceCard( txt: "CHANCEKORT: Du har modtaget aktieudbytte, du modtager derfor " + 1000 + "KR" );
407         player.gui_player.setBalance(player.gui_player.getBalance() + 1000);
408         gui.displayChanceCard();

```

```

409     }
410
411     @private void parkingTicket(Player player) {
412         gui.displayChanceCard( txt: "CHANCEKORT: Du har fået en parkeringsbøde, betal " + 200 + "KR" );
413         player.gui_player.setBalance(player.gui_player.getBalance() - 200);
414         gui.displayChanceCard();
415     }
416
417     @private void carTires(Player player) {
418         gui.displayChanceCard( txt: "CHANCEKORT: Du har købt fire nye dæk til din bil, betal " + 1000 + "KR" );
419         player.gui_player.setBalance(player.gui_player.getBalance() - 1000);
420         gui.displayChanceCard();
421     }
422
423
424     private void checkBalance() {
425         for (int i = 0; i < player.length; i++) {
426             if (player[i].gui_player.getBalance() <= 0) {
427                 gui.showMessage( msg: player[i].gui_player.getName() + " har tabt..." );
428
429                 findWinner();
430
431                 this.isPlaying = false;
432                 gui.close();
433             }
434         }
435     }
436

```

```
437     private void findWinner() {
438         int i;
439         int max = player[0].gui_player.getBalance();
440         String name = player[0].gui_player.getName();
441
442         for (i = 1; i < player.length; i++) {
443             if (player[i].gui_player.getBalance() > max) {
444                 max = player[i].gui_player.getBalance();
445                 name = player[i].gui_player.getName();
446             }
447         }
448         gui.showMessage( msg: name + " har vundet!!!");
449     }
450
451     public void game() {
452         setUpPlayers();
453         while (isPlaying) {
454             this.calculatePlayerTurn();
455         }
456     }
457
458 }
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4
5 public class Brewery extends Field {
6
7     int price;
8     String subText;
9     boolean hasOwner;
10    public int rent;
11
12    public Brewery(String name, String subText, int price, int rent, Color bgColor, Color fgColor) {
13        super(name, bgColor, fgColor);
14        this.subText = subText;
15        this.price = price;
16        hasOwner = false;
17        rent = rent;
18    }
19
20    public void setHasOwner(boolean hasOwner) { this.hasOwner = hasOwner; }
21
22    public boolean isHasOwner() { return hasOwner; }
23
24    public int getRent() { return rent; }
25
26    public void setRent(int rent) { this.rent = rent; }
27
28    public String getSubText() { return subText; }
29
30
31
32
33
34
35
36
```

```
36    public String getSubText() { return subText; }
37
38    public int getPrice() { return price; }
39
40    public void setSubText(String subText) { this.subText = subText; }
41
42
43
44
45
46
47
48
49    }
50
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4
5 public class Chance extends Field {
6
7
8    public Chance(String name, Color bgColor, Color fgColor) { super(name, bgColor, fgColor); }
9
10
11
12
13    }
14
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4
5 public abstract class Field {
6
7     String name;
8     Color bgColor, fgColor;
9
10
11     public Field(String name, Color bgColor, Color fgColor) {
12         this.name = name;
13         this.bgColor = bgColor;
14         this.fgColor = fgColor;
15     }
16
17     public void setName(String name) {
18         this.name = name;
19     }
20
21     public String getName() {
22         return name;
23     }
24
25     public Color getFgColor() { return fgColor; }
26
27     public void setBgColor(Color bgColor) { this.bgColor = bgColor; }
28
29
30     public Color getBgColor() { return bgColor; }
31
32
33     public void setFgColor(Color fgColor) { this.fgColor = fgColor; }
34
35
36     public Class landOnField() { return this.getClass(); }
37
38
39
40
41
42
43
44
45
46
47
48 }
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4
5 public class Fleet extends Field {
6
7     int Price, fleet1, fleet2, fleet3, fleet4;
8     String subText;
9     boolean hasOwner;
10
11    public Fleet(String name, String subText, int Price, Color bgColor, Color fgColor) {
12        super(name, bgColor, fgColor);
13        this.Price = Price;
14        this.subText = subText;
15        hasOwner = false;
16        fleet1 = 500;
17        fleet2 = 1000;
18        fleet3 = 2000;
19        fleet4 = 4000;
20    }
21
22    public void setSubText(String subText) { this.subText = subText; }
23
24    public String getSubText() { return subText; }
25
26    public int getPrice() { return Price; }
27
28    public void setHasOwner(boolean hasOwner) { this.hasOwner = hasOwner; }
```

```
37
38    public boolean isHasOwner() { return hasOwner; }
39
40    public int getFleet1() { return fleet1; }
41
42    public void setFleet1(int fleet1) { this.fleet1 = fleet1; }
43
44    public int getFleet2() { return fleet2; }
45
46    public void setFleet2(int fleet2) { this.fleet2 = fleet2; }
47
48    public int getFleet3() { return fleet3; }
49
50    public void setFleet3(int fleet3) { this.fleet3 = fleet3; }
51
52    public int getFleet4() { return fleet4; }
53
54    public void setFleet4(int fleet4) { this.fleet4 = fleet4; }
55
56    public void setPrice(int price) { Price = price; }
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73    public void setPrice(int price) { Price = price; }
74
75
76
77
78
79 }
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4
5 public class Jail extends Field {
6
7     int jailFee;
8     String subText;
9
10
11     public Jail(String name, Color bgColor, Color fgColor, String subText, int jailFee) {
12         super(name, bgColor, fgColor);
13         this.jailFee = jailFee;
14     }
15
16     public void setJailFee(int jailFee) {
17         this.jailFee = jailFee;
18     }
19
20     public void setSubText(String subText) { this.subText = subText; }
21
22     public int getJailFee() { return jailFee; }
23
24     public String getSubText() { return subText; }
25
26 }
27
28
29 }
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4
5 public class JailVisit extends Field {
6
7     String subText;
8
9     public JailVisit(String name, String subText, Color bgColor, Color fgColor) {
10         super(name, bgColor, fgColor);
11         subText = subText;
12     }
13
14     public String getSubText() { return subText; }
15
16     public void setSubText(String subText) { this.subText = subText; }
17
18 }
19
20
21 }
```

```
1 package com.company.Models.Fields;
2
3
4 import java.awt.*;
5
6 public class Parking extends Field {
7
8     String subText;
9
10    public Parking(String name, String subText, Color bgColor, Color fgColor) {
11        super(name, bgColor, fgColor);
12        this.subText = subText;
13    }
14
15    void setSubText(String subText) { this.subText = subText; }
16
17    String getSubText() { return subText; }
18
19    public void setSubText(String subText) { this.subText = subText; }
20
21    public String getSubText() { return subText; }
22
23}
24
25
26
27
28
29
30
31
32}
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4
5 public class Start extends Field {
6
7     String subText;
8     // Giv 4k når over ny start field = start -> giv player 4k
9     int Profit;
10
11    public Start(String name, Color bgColor, Color fgColor, int Profit, String subText) {
12        super(name, bgColor, fgColor);
13        this.subText = subText;
14        this.Profit = Profit;
15    }
16
17    void setSubText(String subText) {
18        this.subText = subText;
19    }
20
21
22    private void setProfit(int profit) { this.Profit = Profit; }
23
24    String getSubText() { return subText; }
25
26    public int getProfit() { return Profit; }
27
28
29
30
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4 public class Street extends Field {
5
6     int price, rent1;
7     String subText;
8     boolean hasOwner;
9
10    public Street(String name, String subText, int price, Color bgColor, Color fgColor, int rent1) {
11        super(name, bgColor, fgColor);
12        this.subText = subText;
13        this.price = price;
14        this.rent1 = rent1;
15        hasOwner = false;
16    }
17    public boolean getOwner() { return hasOwner; }
18
19    public void setHasOwner(boolean hasOwner) { this.hasOwner = hasOwner; }
20
21    public int getPrice() { return price; }
22
23    public int getRent1() { return rent1; }
24
25    public void setSubText(String subText) { this.subText = subText; }
26
27    public String getSubText() { return subText; }
28
29 }
```

```
1 package com.company.Models.Fields;
2
3 import java.awt.*;
4
5 public class Tax extends Field {
6
7     int taxFee;
8     String subText;
9
10    public Tax(String name, Color bgColor, Color fgColor, int taxFee, String subText) {
11        super(name, bgColor, fgColor);
12        this.taxFee = taxFee;
13        this.subText = subText;
14    }
15
16    public String getSubText() { return subText; }
17
18    public void setSubText(String subText) { this.subText = subText; }
19
20    public int getTaxFee() { return taxFee; }
21
22    public void setTaxFee(int taxFee) { this.taxFee = taxFee; }
23
24 }
```

```

1 package com.company.Models;
2
3 import java.util.Random;
4
5 public class ChanceCard {
6
7     private final int[] chanceCard = new int[]{1, 2, 3, 4, 5, 6};
8     Random random = new Random();
9
10
11    public int randomCard() { return this.random.nextInt( bound: this.chanceCard.length - 1) + 1; }
12
13
14}
15
16

```

```

1 package com.company.Models;
2
3 import java.util.Random;
4
5 public class Die {
6
7     public int diceNumber;
8     private final Random random = new Random();
9     private final int MIN;
10    private final int MAX;
11
12    /* Konstruktør til Die klassen, hvor man skal angive en MIN og MAX værdi af terningen. Fx 1 til 6 for en normal 6
13     * sided terning */
14
15    public Die(int MIN, int MAX) {
16        this.MIN = MIN;
17        this.MAX = MAX;
18    }
19
20    // Metode der slår med en enkelt terning
21    public int rollDice() {
22        diceNumber = random.nextInt((MAX - MIN + 1)) + MIN;
23        return diceNumber;
24    }
25

```

```

26    /* Metode der bruger en eller flere instanser af Die klassen, som ruller med terninger via rollDice() metoden fra
27     * Die klassen. Metoden accepterer flere argumenter. */
28    @
29    public int diceTurn(Die... args) {
30
31        int sum = 0;
32        for (Die arg : args) {
33            arg.rollDice();
34            sum += arg.diceNumber;
35        }
36
37        return sum;
38    }
39
40    // Metode der returner summen af to terninger
41    public static int getSum(int dice1) {
42        return dice1;
43    }
44
45}

```

```
1  package com.company.Models;
2
3  import gui_fields.GUI_Car;
4  import gui_fields.GUI_Player;
5  import gui_main.GUI;
6
7  import java.awt.*;
8
9  public class Player {
10
11     private String name;
12     public int playerPosition = 0;
13     private int ownedBreweries;
14     private int ownedFleets;
15     public GUI_Player gui_player;
16
17     @
18     public Player(GUI gui, Color color) {
19
20         this.name = gui.getUserString("Skriv navn");
21         GUI_Car car = new GUI_Car();
22         car.setPrimaryColor(color);
23
24         gui_player = new GUI_Player(name, balance: 30000, car);
25         gui.addPlayer(gui_player);
26     }
27
28     public String getName() { return name; }
29
30 }
```

```
31
32     public void setOwnedBreweries() { this.ownedBreweries += 1; }
33
34     public int getOwnedFleets() { return ownedFleets; }
35
36     public void setOwnedFleets() { this.ownedFleets += 1; }
37
38     public GUI_Player getGui_player() {
39         return gui_player;
40     }
41
42 }
```

```

1 package com.company.Views;
2
3 import com.company.Models.Fields.*;
4 import java.awt.*;
5
6 public class BoardGUI {
7     public static Field[] fields = {
8         new Start( name: "START", Color.red, Color.BLACK, Profit: 4000, subText: "Modtag kr. 4000"),
9         new Street( name: "Rødovrevej", subText: "1.200KR", price: 1200, Color.blue, Color.BLACK, rent1: 50),
10        new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
11        new Street( name: "Hvidovrevej", subText: "1.200KR", price: 1200, Color.blue, Color.BLACK, rent1: 50),
12        new Tax( name: "Skat", Color.cyan, Color.BLACK, taxFee: 4000, subText: "Betal 4000KR"),
13        new Fleet( name: "Scandilines", subText: "Farge", Price: 4000, Color.BLUE, Color.BLACK),
14        new Street( name: "Rodskildevej", subText: "2.000KR", price: 2000, Color.ORANGE, Color.BLACK, rent1: 100),
15        new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
16        new Street( name: "Valby Langgade", subText: "2.000KR", price: 2000, Color.ORANGE, Color.BLACK, rent1: 100),
17        new Street( name: "Allégade", subText: "2.400KR", price: 2400, Color.ORANGE, Color.BLACK, rent1: 150),
18        new JailVisit( name: "Fængsel", subText: "På besøg", Color.gray, Color.BLACK),
19        new Street( name: "Frederiksberg Allé", subText: "2.800KR", price: 2800, Color.GREEN, Color.BLACK, rent1: 200),
20        new Brewery( name: "Turborg Squash", subText: "3.000KR", price: 3000, rent: 100, Color.CYAN, Color.BLACK),
21        new Street( name: "Büllowsvej", subText: "2.800KR", price: 2800, Color.GREEN, Color.BLACK, rent1: 200),
22        new Street( name: "Gl.Kongevej", subText: "3.200KR", price: 3200, Color.GREEN, Color.BLACK, rent1: 250),
23        new Fleet( name: "Mols-Linien", subText: "4.000kr", Price: 4000, Color.RED, Color.BLACK),
24        new Street( name: "Bernstorftsvej", subText: "3.600KR", price: 3600, Color.GRAY, Color.BLACK, rent1: 300),
25        new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
26        new Street( name: "Hellerupvej", subText: "3.600KR", price: 3600, Color.GRAY, Color.BLACK, rent1: 300),
27        new Street( name: "Strandvejen", subText: "4000KR", price: 4000, Color.GRAY, Color.BLACK, rent1: 350),
28        new Parking( name: "Parking", subText: "Parking", Color.BLUE, Color.BLACK),
29
30        new Street( name: "Trianglen", subText: "4.400KR", price: 4400, Color.RED, Color.BLACK, rent1: 350),
31        new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
32        new Street( name: "Østerbrogade", subText: "4.400KR", price: 4400, Color.RED, Color.BLACK, rent1: 350),
33        new Street( name: "Grønningen", subText: "4.800KR", price: 4800, Color.RED, Color.BLACK, rent1: 400),
34        new Fleet( name: "Scandilines", subText: "4.000KR", Price: 4000, Color.BLUE, Color.BLACK),
35        new Street( name: "Bredgade", subText: "5.200KR", price: 5200, Color.WHITE, Color.BLACK, rent1: 450),
36        new Street( name: "Kgs.Nytorg", subText: "5.200KR", price: 5200, Color.white, Color.BLACK, rent1: 450),
37        new Brewery( name: "Coca cola", subText: "3.000KR", price: 3000, rent: 100, Color.RED, Color.BLACK),
38        new Street( name: "Østergade", subText: "5.600KR", price: 5600, Color.white, Color.BLACK, rent1: 500),
39        new Jail( name: "Fængsel", Color.gray, Color.black, subText: "De fængsles", jailFee: 1000),
40        new Street( name: "Amagertorv", subText: "6.000KR", price: 6000, Color.YELLOW, Color.BLACK, rent1: 550),
41        new Street( name: "Vimmelskattet", subText: "6.000KR", price: 6000, Color.YELLOW, Color.BLACK, rent1: 550),
42        new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
43        new Street( name: "Nygade", subText: "6.400KR", price: 6400, Color.YELLOW, Color.BLACK, rent1: 600),
44        new Fleet( name: "Scandilines", subText: "4.000KR", Price: 4000, Color.BLUE, Color.BLACK),
45        new Chance( name: "Prøv lykken", Color.BLACK, Color.GREEN),
46        new Street( name: "Frederiksbergsgade", subText: "7000KR", price: 7000, Color.MAGENTA, Color.BLACK, rent1: 700),
47        new Tax( name: "Ekstra ordinær Statskat", Color.CYAN, Color.BLACK, taxFee: 2000, subText: "Betal 2000KR"),
48        new Street( name: "Rådhuspladsen", subText: "8.000KR", price: 8000, Color.MAGENTA, Color.BLACK, rent1: 1000),
49    };
50 }
51

```

```
1 package com.company;
2
3 import com.company.Controllers.GameController;
4 import com.company.Views.BoardGUI;
5 import com.company.Controllers.helper.ConvertHelper;
6 import gui_main.GUI;
7
8 public class Main {
9     public static void main(String[] args) {
10         GameController game = new GameController(new GUI(ConvertHelper.guiFieldConvert(BoardGUI.fields)));
11         game.game();
12     }
13 }
14
15 }
```

11 LITTERATUR- OG KILDEFORTEGNELSER

Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Pearson.