Table of Contents

Class 2
- GameController 2
- MoveController 11
- BalanceCard 12
- Deck 13
- FamilyCard 15
- FreeJailCard 16
- MoveCard 17
- MoveFleetCard 17
- Brewery 18
- Chance 19
- Fieldlist 21
- Fleet 22
- GoToPrison 23
- Parking 25
- Plot 26
- BankAccount 28
- Dicecup 29
- Die 30
- Player 31
- Alltests 32
- JUnitTestBrewery 34
- JUnitTestDie 35
- JUnitTestFieldlist 36
- JUnitTestFleet 37
- JUnitTestPlayer 38
- JUnitTestStartField 39
- JUnitTestTax 39
- TestData 40
- FakeOutputFalse 42
- FakeOutputTrue 45
- Language 49
- Output 55

Class

GameController

```
package controller;

import java.util.ArrayList;
import java.util.List;

import desktop_resources.GUI;
import model.Dicecup;
import model.Player;
import model.cards.Deck;
import model.fields.Field;
import model.fields.Fieldlist;
import model.fields.Ownable;
import model.fields.Plot;
import view.Language;
import view.Out;
import view.Output;

/*
 * Team 18 - CDIO 3
 * DTU
 * Collaborators:  KasperLeiszner, Bijan Negari, Helene Zgaya, Frederik von Scholten and
Troels Lund

/*
 * Class wrote by: Troels Lund and Kasper Leiszner
 */

public class Gamecontroller
{
	private static Player[] player;

	public static  Player[] getPlayer() {
		return player;
	}

	private Dicecup cup = new Dicecup();
	private Out out;
	private boolean endGame;
	private MoveController movecontroller = new MoveController();

	public static void main(String[] args)
	{
		new Gamecontroller().setup(); //Opretter objekt af gamecontroller og kalder
setup
	}

	public Gamecontroller()
	{
		this(new Output()); //Kalder kontruktøren nedenunder.
	}
	public Gamecontroller(Out out) //Ved test bruges kun denne konstruktør og ikke den
ovenstående.
	{
		this.out = out; //
		new Fieldlist(out);
		new Deck(out);
	}

	public void setup(){ // Sætter spillet op, opretter spillere
		player = addPlayer();
		runGame();
	}

	public void runGame()
	{ // Kører spil med turskift
		while(!endGame)
		{
			for( int i = 0 ; i < player.length ; i++ )
			{
				Player p = player[i];

				if(!p.getBankruptStatus())
				{
					out.msgGUI(Language.whoseTurn(p));
					turn(p);
				}
			}
		}
	}

	public void turn(Player p)
	{ //kører en tur for den aktuelle spiller


		if(p.isJailed()){
			prisonAction(p);

		}
		else{
			out.rollDiceText();
			cup.roll();

			int amountOfMoves = cup.getSum();

			out.setGUIDice(cup.getDie1().getValue(), cup.getDie2().getValue());

			out.removeCar(p);
			movecontroller.movePlayer(amountOfMoves, p);
			out.setcar(p);
			out.landMSG(p);

			Field f = Fieldlist.getFields()[p.getPlayerPos()];
			f.landOn(p, out); // Kalder landOn for spillerens position i
feltlistens array
```

```
            for (int i = 0; i < player.length; i++)
            {
                out.setGUIBalance(player[i]);
            }

            goBankrupt();
            winner();
            checkPlots(p, out);

//          if(){
//
//          }

            if(cup.getDie1().getValue() == cup.getDie2().getValue()){

                out.msgGUI(Language.extraTurn(p));

                turn(p);
            }
        }

        System.out.println();
    }

    public void prisonAction(Player p){ // metode til at håndtere vis man er i fængsel

        String[] option;

        String a = "Køb dig fri for 50kr";
        String b = "Slå dig fri";
        String c = "Brug dit chancekort";

        if(!(p.getJailcards().isEmpty())){ // Hvis du har et jail-card, får du tre
mulige menuer:
            option = new String[3];
            option[0] = a;
            option[1] = b;
            option[2] = c;
        }
        else{
            option = new String[2];
            option[0] = a;
            option[1] = b;
        }

        switch (out.Jailaction(p, option)){ // hvad skal der ske når man vægler en
af de tre mulighder?:
        case "Køb dig fri for 50kr":
            p.getAccount().withdraw(50);
            p.setJailed(false);
            turn(p);

            break;
```

```
			case "Slå dig fri":
				out.msgGUI("Slå to ens for at komme fri!");
				out.rollDiceText();
				cup.roll();
				out.setGUIDice(cup.getDie1().getValue(), cup.getDie2().getValue());
				if(cup.getDie1().getValue() == cup.getDie2().getValue()){
					p.setJailed(false);
					turn(p);	// hvis en spiller slår 2 ens, får han sin tur 
igen, så rykker han fra fængelsfeltet, og kan købe grunde som normalt.
				}
				break;

			case "Brug dit chancekort":
				p.setJailed(false);
				p.getJailcards().remove(0).setOwner(null);

				turn(p);		// hvis en spiller bruger jail-card, får han 
sin tur igen, så rykker han fra fængelsfeltet, og kan købe grunde som normalt.
				break;
			}
	}

	//	public void movePlayer(Player p, int amountOfMoves)
	//	{
	//		if(p.getPlayerPos() + amountOfMoves > Fieldlist.getFields().length)
	//		{
	//			p.setPlayerPos((p.getPlayerPos() + amountOfMoves)-
Fieldlist.getFields().length); //Hvis antal ryk og spillerens position overskrider 
feltlistens længde, trækkes den fra
	//			p.getAccount().addSum(4000); //Start bonus
	//		}
	//		else
	//		{
	//			p.setPlayerPos(p.getPlayerPos() + amountOfMoves);
	//		}
	//	}

	public Player[] addPlayer()
	{
		Player[] player = new Player[out.howManyPlayers()]; // Opretter antal 
spillere fra input i GUI

		for( int i = 0 ; i < player.length ; i++ )
		{
			String name = GUI.getUserString(Language.getNameOfPlayer() + " " + (i 
+ 1)); //input navn fra GUI(skal laves i Output)

			player[i] = new Player(name);
			player[i].setPlayerPos(0);
			out.addPlayersToGUI(player[i]);
		}

		return player; //
	}
	public void resetOwnedFields(Player p)
	{
		if(p.getBankruptStatus() == true)
		{
			for(int i = 0; i < Fieldlist.getFields().length;i++)
			{
				if(Fieldlist.getFields()[i] instanceof Ownable)
				{
					if( ((Ownable) Fieldlist.getFields()[i]).getOwner() ==
p )
					{
						((Ownable)
Fieldlist.getFields()[i]).setOwner(null);
						out.removeOwner(p.getPlayerPos()+1);
					}
				}
			}
		}
	}

	public void goBankrupt()
	{
		for (int i = 0; i < player.length; i++)
		{
			if(player[i].getAccount().getSum() <= 0)
			{
				player[i].setBankrupt(true);
				resetOwnedFields(player[i]);
				GUI.removeCar(player[i].getPlayerPos(), player[i].getName());
			}
		}
	}

	public void winner()
	{

		int playersAlive = 0;

		for (int i = 0; i < player.length; i++)
		{
			Player p = player[i];
			boolean bankrupt = p.getBankruptStatus(); // Tjekker om spillere er
bankrupt
			if(!bankrupt) playersAlive++; //ligger en til hver gang spilleren
ikke er bankrupt
		}

		if(playersAlive == 1)
		{

			for (int i = 0; i < player.length; i++)
			{ // Tjekker HVILKEN spiller der er tilbage

				Player p = player[i];
				boolean bankrupt = p.getBankruptStatus();
```

```
                        if(!bankrupt)
                        {
                                out.msgGUI(Language.getWinnerText() + " " +
p.getName()); //Prøver at holde pause i 10 sekunder efter vinder er fundet. Ellers laver
den exception så programmet ikke crasher
                                out.closeGame();
                                endGame = true;
                        }
                }
        }
    }

    public Player[] getPlayerArray(){
            return player;
    }

    public void checkPlots(Player p, Out out){

            int[] gruppeNumre = generateBoughtFieldArray(p); // udfylder gruppenumre med
data om havd spilleren ejer
            List<Field> flist = addPlotsToList(gruppeNumre); // tjekker om spiller ejer
alle grunde i en gruppe, og giver tilbage en liste med fields som han kan bygge på.


            if(flist.size() < 1){ // vis der ikke er nogle grunde i flist - STOP her.
                    return;
            }

            if(out.shopOrNot()){
                    if(out.sellOrBuy()){

                            String result = out.whereToBuild(generateNameArray(flist)); //
spørg i GUI

                            int index = findIndexOfField(result); // finder index for
placering af field der skal bygges på.

                            buyHouse(p,result,index);

                    }
                    else{

                            String result = out.whereToSell(generateNameArraySell(flist));

                            int index = findIndexOfField(result);

                            sellHouse(p,result,index);

                    }
            }

    }
```

```
	private int[] generateBoughtFieldArray(Player p){

		int[] gruppeNumre = new int[8];

		for(int i = 0; i < Fieldlist.getFields().length; i++)	// for-loop der 
kører alle felter igennem.
		{
			if(Fieldlist.getFields()[i] instanceof Plot)		// Sorterer 
alle felter der IKKE er plot fra. Vi tjekker altså om det specifik felt er et Plot-felt.
			{
				Plot plot = (Plot) Fieldlist.getFields()[i];	// caster 
feltet vi ved er et Plot til et Plot ås vi kan få adgang til GroupNumber
				if(plot.getOwner() != null && plot.getOwner().equals(p)){ // 
tjekker at plot har en owner og derved ikke er Null og der næst at owneren er den samme som 
den spiller osm har tur
					//TODO
					gruppeNumre[plot.getGroupNumber()-1] += 1;  // 
tilføjer at du har købt en den type grund til array.
				}


			}
		}
		return gruppeNumre;
	}

	private List<Field>  addPlotsToList(int[] gruppeNumre){

		List<Field> flist = new ArrayList<>();
		int[] maxPlot = {2, 3, 3, 3, 3, 3, 3, 2};
		boolean[] canBuild = new boolean[8];

		for (int i = 0; i < maxPlot.length; i++) { // du kan bygge vis antalet af 
grunde inde for en type er det samme som hvad du max kan få.
			if(gruppeNumre[i] == maxPlot[i]){
				canBuild[i] = true;
			}
		}

		for (int i = 0; i < canBuild.length; i++) { // adder gruppe til mulig 
byggegrunde.
			if(canBuild[i]){

				//TODO
				flist.addAll(getGroup(i+1));
			}
		}

		return flist;

	}

	private String[] generateNameArray(List<Field> flist){

		String[] FieldNames = new String[flist.size()]; // opretter array.
		for (int i = 0; i < flist.size(); i++) { // sætter navne på felter ind i
Navne-array
			FieldNames[i] = flist.get(i).getName();
		}
		return FieldNames;
	}

	public List<Field> getGroup(int x){
		List<Field> flist = new ArrayList<>();
		for (Field f : Fieldlist.getFields()) {
			if(f instanceof Plot ){
				Plot p = (Plot) f;
				if(p.getGroupNumber() == x){
					flist.add(p);
				}
			}
		}
		return flist;
	}

	private int findIndexOfField(String result){
		for (int i = 0; i < Fieldlist.getFields().length; i++) {
			if(Fieldlist.getFields()[i].getName().equals(result)) {
				return i;
			}
		}
		return 0;
	}

	private String[] generateNameArraySell(List<Field> flist){

		String[] FieldNames = new String[flist.size()]; // opretter array.


		for (int i = 0; i < flist.size(); i++) { // sætter navne på felter ind i
Navne-array
			if(flist.get(i) instanceof Plot){
				Plot p =(Plot) flist.get(i);
				if(p.getHousecount() > 0){
					FieldNames[i] = flist.get(i).getName();

				}
			}
		}
		return FieldNames;
	}

	private void buyHouse(Player p, String result, int index){

			Field f = Fieldlist.getFields()[index];
			if(f.getName().equals(result)){
				if(f instanceof Plot ){
					Plot plotCast =(Plot) f;
					plotCast.upgradePlot();
```

```
                                    if(plotCast.getHousecount() >= 5){ // bygger hotel vis
der er 5 huse.
                                           out.BuildHotel(index+1, true);
                                           p.getAccount().withdraw(200);
                                    }
                                    else{
                                           out.BuildHouse(index+1,
plotCast.getHousecount()); // bygger huse vis der er mindre end 5 huse.
                                           p.getAccount().withdraw(200);
                                    }

                             }

                      }
       }


private void sellHouse(Player p, String result, int index){

                      Field f = Fieldlist.getFields()[index];
                      if(f.getName().equals(result)){
                             if(f instanceof Plot ){
                                    Plot plotCast =(Plot) f;
                                    plotCast.downgradePlot();
                                    if(plotCast.getHousecount() >= 5){ // bygger hotel vis
der er 5 huse.
                                           out.BuildHotel(index, false);
                                           p.getAccount().addSum(200);
                                           out.BuildHouse(index+1,
plotCast.getHousecount());
                                    }
                                    else{
                                           out.BuildHouse(index+1,
plotCast.getHousecount()); // bygger huse vis der er mindre end 5 huse.
                                           p.getAccount().addSum(200);
                                    }

                             }

                      }


       }
}
```

MoveController

```
package controller;

import model.Player;

import model.fields.Fieldlist;

public class MoveController {


	public void movePlayer(int amountOfMoves, Player player)
	{
		if((player.getPlayerPos() + amountOfMoves) >= Fieldlist.getFields().length)
{

			player.setPlayerPos((player.getPlayerPos() + amountOfMoves) - 
Fieldlist.getFields().length);
			player.getAccount().addSum(200);
		}
		else
		{
			player.setPlayerPos(amountOfMoves + player.getPlayerPos());
		}
	}
}
```

BalanceCard

```
package model.cards;

import model.Player;
import view.Out;

public class BalanceCard extends Card {

	int BalanceModifier;

	public BalanceCard(String s, int balanceModifier,Out out) {
		super(s, out);
		BalanceModifier = balanceModifier;
	}

	@Override
	public void doCard(Player p) {
		out.CardsOut(description);
		if(BalanceModifier <0){
			p.getAccount().withdraw(BalanceModifier * (-1));
		}
		else{
		p.getAccount().addSum(BalanceModifier);
		}
		out.setGUIBalance(p);
	}



}
```

```
package model.cards;

import model.Player;
import view.Out;

public abstract class Card  {

	protected String description;
	protected Out out;
	public Card(String description, Out out) {
		this.description = description;
		this.out = out;
	}

	public abstract void doCard(Player p);

	@Override
	public String toString() {
		return description;
	}


}
```

Deck

```
package model.cards;

import java.util.Arrays;
import java.util.Random;

import view.Language;
import view.Out;

public class Deck {

	private static int cardCount = 0;
	private static Card[] cards;
	public Out out;
	private String[] des = Language.getCardDescriptions();

	public Deck(Out output){
		this.out = output;

		cards = new Card[]{
//			new BalanceCard(des[0],50, 125, out),		// betal 50 pr. 
hus og 125 pr. hotel.
			new MoveCard(des[1], 11, true, out),		// teleport til 
Frederiksberg Allé.
			new FamilyCard(des[2], 25, out),		// tager 25 dask 
fra hver spiller og giver til spilleren der trækker kortet.
			new MoveCard(des[3], -3, false, out),		// ryk 3 felter 
tilbage
			new BalanceCard(des[4], 50, out),		// modtag 50 dask.
			new BalanceCard(des[5], -15, out),		// modtag 25 dask.
			new BalanceCard(des[6], -10, out),		// betal 10 dask.
			new BalanceCard(des[7], 100, out),		// modtag 100 
dask.
			new BalanceCard(des[8], 100, out),		// modtag 100 
dask.
```

```
                    new MoveCard(des[9],0, true, out),                // ryk frem til
START
                    new MoveCard(des[10],0, true, out),               // ryk frem til
START
                    new FamilyCard(des[11], 25, out),                 // modtag 25 dask
fra hver spiller
                    new MoveCard(des[12], 39, true, out),             // tag ind på
rådhuspladsen
                    new MoveCard(des[13], 5, true , out),             // ryk til
øresundsredderiet
                    new BalanceCard(des[14], -50, out),               // betal 50 dask
                    new BalanceCard(des[15], -20, out),               // betal 20 dasko
                    new BalanceCard(des[16], -10, out),               // betal 10 dask
                    new MoveFleetCard(des[17], false, out),           // tag med
nærmeste redderi :)
                    new MoveCard(des[18], 30, true, out),             // gå i fængsel
                    new MoveCard(des[19], 30, true, out),             // gå i fængsel
bra
       //           new BalanceCard(des[20], 25, 100, out),     // betal 25 pr. hus og
100 pr. hotæl
                    new BalanceCard(des[21], 50, out),                // modtag 50 dask
                    new BalanceCard(des[22], 50, out),                // modtag 50 dask
                    new MoveFleetCard(des[23], true, out),            // ryk til
nærmeste redderi og betal double husleje
                    new MoveCard(des[24], 19, true, out),             // Ryk frem til
Strandvejen
                    new BalanceCard(des[25], -50, out),               // betal dask 50
                    new BalanceCard(des[26], 50, out),                // modtag 50 dask
                    new BalanceCard(des[27], 25, out),                // modtag 25 dask
                    new BalanceCard(des[28], 25, out),                // modtag 25 dask
                    new MoveCard(des[29], 32, true, out),             // ryk til
Vimmelskaftet
                    new MoveCard(des[30], -3, false, out),            // ryk tre felter
tilbage
                    new MoveCard(des[31], -3, false, out),            // ryk tre felter
tilbage
                    new FreeJailCard(des[32], out),                           //
FreeJailCard
                    new FreeJailCard(des[33], out),                           //
FreeJailCard
                    new BalanceCard(des[34], -150, out),        // betal 150 dask
                    new BalanceCard(des[35], 50, out),                // modtag 50 dask
                    new BalanceCard(des[36], -10, out),               // betal 10 dask
                    new MoveCard(des[37], 24, true, out),             // ryk til
grønningen
                    new BalanceCard(des[38], 150, out),               // modtag 150 dask
                    new FamilyCard(des[39], 25, out),                 // fra hver
spiller, modtag 25 dask
                    new BalanceCard(des[40], -50, out),               // betal 50 dask i
bøde bro
                    new BalanceCard(des[41], 10, out),                // betal 10 dask
                    new BalanceCard(des[42], 100, out),               // modtag dask 100
af banken
              };
              shuffleArray();
       }
```

```
	public static void shuffleArray(){
		Random rnd = new Random();
		for (int i = cards.length - 1; i > 0; i--)
		{
			int index = rnd.nextInt(i + 1);
			// Simple swap
			Card a = cards[index];
			cards[index] = cards[i];
			cards[i] = a;
		}

	}



	public static int getCardCount() {
		return cardCount;
	}

	public static void setCardCount(int cardCount) {
		Deck.cardCount = cardCount;
	}

	public static Card[] getCards() {
		return cards;
	}

	public static void printCards(){
		System.out.println(Arrays.toString(Deck.cards));
	}

}
```

FamilyCard

```
package model.cards;

import controller.Gamecontroller;
import model.Player;
import view.Out;

public class FamilyCard extends Card  {

	int gift;

	public FamilyCard(String description,int gift, Out out) {
		super(description, out);
		this.gift = gift;
	}

	@Override
	public void doCard(Player p) {
		out.CardsOut(description);
		for (int i = 0; i < Gamecontroller.getPlayer().length; i++) {
			Gamecontroller.getPlayer()[i].getAccount().withdraw(gift);
			p.getAccount().addSum(gift);
			System.out.println(p.getName() + "got " + gift + " kr form " + 
Gamecontroller.getPlayer()[i].getName());
			out.setGUIBalance(Gamecontroller.getPlayer()[i]);
		}
		out.setGUIBalance(p);
	}




}
```

FreeJailCard

```
package model.cards;

import model.Player;
import view.Out;

public class FreeJailCard extends Card {

	private Player owner;

	public FreeJailCard(String des, Out out) {
		super(des, out);
		this.out = out;
	}

	public void doCard(Player p){
		out.CardsOut(description);
		setOwner(p);
		p.getJailcards().add(this);
	}

	public Player getOwner() {
		return owner;
	}

	public void setOwner(Player owner) {
		this.owner = owner;
	}





}
```

MoveCard

```
package model.cards;

import controller.MoveController;
import model.Player;
import model.fields.Fieldlist;
import view.Out;

public class MoveCard extends Card {

        private int move;
        private boolean teleport;     //teleport or move care

        public MoveCard(String description, int moves, boolean teleport, Out out) {
                super(description, out);
                this.move = moves;
                this.teleport = teleport;

        }

        public void doCard(Player p){
                out.CardsOut(description);
                out.removeCar(p);
                if(teleport){
                        if(move == 24 && p.getPlayerPos() > 24){
                                p.getAccount().addSum(200);
                        }
                        if(move == 30){
                                Fieldlist.getFields()[30].landOn(p, out);
                        }
                        p.setPlayerPos(move);
                }
                else{
                        new MoveController().movePlayer(move, p);
                }

                out.setcar(p);
        }
}
```

MoveFleetCard

```
package model.cards;

import model.Player;
import model.fields.Field;
import model.fields.Fieldlist;
import model.fields.Fleet;
import view.Out;
```

```
public class MoveFleetCard extends Card {

	private boolean doubleRent;

	public MoveFleetCard(String description, boolean doubleRent, Out out) {
		super(description, out);
		this.doubleRent = doubleRent;
	}

	@Override
	public void doCard(Player p) {
		int pos = 0;
		out.removeCar(p);
		out.CardsOut(description);
		for (int i = 0; i <= Fieldlist.getFields().length - 1; i++) {
			int index = (i + p.getPlayerPos()) % 40;
			Field f = Fieldlist.getFields()[index];
			if (f instanceof Fleet) {
				Fleet fleet = (Fleet)f;

				if ((i + p.getPlayerPos()) > Fieldlist.getFields().length) {
					p.getAccount().addSum(200);
				}

				pos = (i + p.getPlayerPos()) % 40;

				if(fleet.getOwner() != null && doubleRent){

	p.getAccount().withdraw(fleet.getRent(fleet.getOwner()) * 2);

	fleet.getOwner().getAccount().addSum(fleet.getRent(fleet.getOwner()) * 2);
				}

				break;
			}

		}
		p.setPlayerPos(pos);
		out.setcar(p);
		out.setGUIBalance(p);
	}
}
```

Brewery

```
package model.fields;

import model.Dicecup;
import model.Player;
import view.Out;

public class Brewery extends Ownable
```

```
// Brewery klassen arver (extender) fra Ownable
{
	private int baseRent;
	Dicecup cup = new Dicecup();

	public Brewery(String name, String description, int price, int baseRent, Out out) {
		super(name, description, price, out);
		this.baseRent = baseRent;
		// this.baseRent sætter attributen der blev oprettet i klassen lig..
	}

	@Override
	public int getRent(Player p)
	// baseRent er 4. Så man ganger 4 med antal øjne. Men har man begge brewery
	// bliver baseRent 10. Derfor ligger vi 6 til i "if".

	{ //
		if (this.getOwner().getBreweryCount() > 1)
		// hvis denne spiller (Player p) har flere end et brewery
		{
			baseRent += 6;
			// kan også skrives: baseRent = baseRent + 6
		}

		// opretter et nyt Dicecup objekt
		out.rollDiceText();
		// besked i GUI (Kast terningerne)
		cup.roll();
		// kalder metoden rool i cup;
		out.setGUIDice(cup.getDie1().getValue(), cup.getDie2().getValue());
		// viser terningernes værdier i GUI
		int sum = cup.getSum();
		// kalder kalder metoden getSum i cup og sætter sum lig return
		int rent = sum * baseRent;
		// sætter rent lig terningernes sum gange baseRent

		return rent;
	}

	public Dicecup getDiceCup()
	{
		return cup;
	}

}
```

Chance

```
package model.fields;

import model.Player;
import model.cards.Deck;
import view.Out;
```

```
public class Chance extends Field
{


	public Chance(String name, String description, Out out)
	{
		super(name, description, out);
	}

	public void landOn(Player p, Out o)
	{
		Deck.getCards()[Deck.getCardCount()].doCard(p); // tag kort i bunken.
		Deck.setCardCount((1+Deck.getCardCount()) % Deck.getCards().length);
	}

	@Override
	public int getValue() {

		return 0;
	}










}

package model.fields;

import model.Player;
import view.Out;

public abstract class Field
{
	private String name = "None";
	private String description = "";
	protected Out out;

	public Field(String name, String description, Out o)
	{
		this.name = name;
		this.description = description;
		out = o;
	}

	public String getName()
	{
		return name;
	}

	public String getDescription()
	{
		return description;
	}

	public abstract void landOn(Player p, Out o);

	public String toString()
	{
		return "Name: " + name + "\n" +
				"Description: " + description;
	}

	public abstract int getValue();


}
```

Fieldlist

```
package model.fields;

import view.Language;
import view.Out;

public class Fieldlist
{
	//an array of the fieldlist
	private static model.fields.Field[] fields;
	private String[] FieldNames = Language.getFieldNames();
	private String[] FieldDescription = Language.getFieldDecription();

	//list of fields
	public Fieldlist(Out out)
	{
		fields = new model.fields.Field[]
			{
				new Parking(FieldNames[0], FieldDescription[0], 0, out),
				new Plot(FieldNames[1], FieldDescription[1], 60, 2, out, 1),
				new Chance(FieldNames[2], FieldDescription[1], out),
				new Plot(FieldNames[3], FieldDescription[1], 60, 4, out, 1),
				new Tax(FieldNames[4], FieldDescription[1], 200, 10, out),
				new Fleet(FieldNames[5], FieldDescription[1], 200, 25,out),
				new Plot(FieldNames[6], FieldDescription[1], 100,6,out, 2),
				new Chance(FieldNames[7], FieldDescription[1], out),
				new Plot(FieldNames[8], FieldDescription[1], 100, 6,out, 2),
				new Plot(FieldNames[9], FieldDescription[1], 120, 8,out, 2),
				new Parking(FieldNames[10], FieldDescription[1], 0, out),
				new Plot(FieldNames[11], FieldDescription[1], 140, 10,out, 3),
				new Brewery(FieldNames[12], FieldDescription[2], 150, 4,out),
                                new Plot(FieldNames[13], FieldDescription[2], 140, 10,out, 3),
                                new Plot(FieldNames[14], FieldDescription[3], 160, 12,out, 3),
                                new Fleet(FieldNames[15], FieldDescription[3], 200, 25,out),
                                new Plot(FieldNames[16], FieldDescription[4], 180, 14,out, 4),
                                new Chance(FieldNames[17], FieldDescription[4], out),
                                new Plot(FieldNames[18], FieldDescription[5], 180, 14,out, 4),
                                new Plot(FieldNames[19], FieldDescription[5], 200, 16,out, 4),
                                new Parking(FieldNames[20], FieldDescription[5], 0, out),
                                new Plot(FieldNames[21], FieldDescription[5], 220, 18,out, 5),
                                new Chance(FieldNames[22], FieldDescription[5] ,out),
                                new Plot(FieldNames[23], FieldDescription[5], 220, 18,out, 5),
                                new Plot(FieldNames[24], FieldDescription[5], 240, 20,out, 5),
                                new Fleet(FieldNames[25], FieldDescription[5], 200, 25,out),
                                new Plot(FieldNames[26], FieldDescription[5], 260, 22,out, 6),
                                new Plot(FieldNames[27], FieldDescription[5], 260, 22,out, 6),
                                new Brewery(FieldNames[28], FieldDescription[5], 150, 4,out),
                                new Plot(FieldNames[29], FieldDescription[5], 280, 24,out, 6),
                                new GoToPrison(FieldNames[30], FieldDescription[5],out),
                                new Plot(FieldNames[31], FieldDescription[5], 300, 26,out, 7),
                                new Plot(FieldNames[32], FieldDescription[5], 300, 26,out, 7),
                                new Chance(FieldNames[33], FieldDescription[5], out),
                                new Plot(FieldNames[34], FieldDescription[5], 320, 28,out, 7),
                                new Fleet(FieldNames[35], FieldDescription[5], 200, 25,out),
                                new Chance(FieldNames[36], FieldDescription[5], out),
                                new Plot(FieldNames[37], FieldDescription[5], 350, 35,out, 8),
                                new Tax(FieldNames[38], FieldDescription[5], 100, 0, out),
                                new Plot(FieldNames[39], FieldDescription[5], 400, 50,out, 8),
                        };
        }

        public static model.fields.Field[] getFields()
        {
                return fields;
        }

}
```

Fleet

```
package model.fields;

import model.Player;
import view.Out;

public class Fleet extends Ownable
// Fleet arver (extender) fra Ownable
{

        private int baseRent;
        // this.baseRent

        public Fleet(String name, String description, int price, int baseRent, Out out) {
                super(name, description, price, out);
```

```
		this.baseRent = baseRent;
		// this.baseRent sætter attributten der blev oprettet i klassen lig..
	}

	@Override
	public int getRent(Player p) // get metode
	{
		return (int) (baseRent * Math.pow(2, (p.getFleetCount() - 1)));
		// returnere Fleet-feltets rent vha. formlen
	}
}
```

GoToPrison

```
package model.fields;

import model.Player;
import view.Out;

public class GoToPrison extends Field
{
	public GoToPrison(String name, String decsription, Out out)
	{
		super(name, decsription, out);
	}

	public void landOn(Player p, Out o)
	{
		p.setJailed(true);
		out.removeCar(p);
		p.setPlayerPos(10);
		out.setcar(p);
	}

	public int getValue()
	{
		return 999999999;
	}
}

package model.fields;

import model.Player;
import view.Out;

public abstract class Ownable extends Field
{

	protected int price;
	private Player owner;
	boolean wantToBuy;
```

```
public Ownable (String name, String description, int price, Out o)
{
	super(name, description, o);
	this.price = price;
}

@Override
public void landOn(Player p, Out o)
{
	if(p.getAccount().getSum()>=price && owner == null) // can buy
	{
		wantToBuy = o.shopField(price, p);

		if(wantToBuy)
		{
			o.setColor(p);
			setOwner(p);

			if(this instanceof Brewery)
			{
				p.setBreweryCount(p.getBreweryCount() + 1);
			}

			if(this instanceof Fleet)
			{
				p.setFleetCount(p.getFleetCount()+1);
			}

			p.getAccount().withdraw(price);
			o.verificationOfPurchase();
		}
	}
	else if(p.getAccount().getSum() < price && owner == null) // cant affort
	{
		o.msgGUI("Du har ikke nok penge til at købe dette felt");;
	}
	else if(p.getAccount().getSum()>=price && owner == null && !wantToBuy)
//Player don't want to buy
	{
		o.deniedPurchase();
	}
	else if(owner != null && owner != p) // is owned
	{
		int rent = getRent(owner);

		// Pay rent
		p.getAccount().withdraw(rent);
		owner.getAccount().addSum(rent);
		o.payedRent(p, rent);
	}
	else	//It's your own field
	{
		o.msgGUI("Dette er din egen grund");
		System.out.println("It's your own field");
	}
}
```

```
	public abstract int getRent(Player p);

	public int getValue()
	{
		return price;
	}

	public void setOwner(Player owner){
		this.owner = owner;
	}

	public Player getOwner(){
		return owner;
	}

}
```

Parking

```
package model.fields;

import model.Player;
import view.Out;

public class Parking extends Field {
	// Parking-klassen arver (extender) fra Field-klassen

	private int bonus;

	public Parking(String name, String description, int bonus, Out out) {
		super(name, description, out);
		this.bonus = bonus;
		// this.bonus sætter attributen der blev oprettet i klassen lig..
	}

	@Override
	public void landOn(Player p, Out o) // skal bruge parametrene Player og Out
	{
		p.getAccount().addSum(bonus);
		// kalder addSum i Player p's account der lægger (bonus) til dit account

	}

	public int getValue() // get metode returnere bonus
	{
		return bonus;
	}

}
```

Plot

```
package model.fields;

import model.Player;
import view.Out;

public class Plot extends Ownable
{
	private int groupNumber;
	private int rent;
	private int housecount = 0;

	public Plot(String name, String description, int price, int rent, Out out, int
groupNumber) {
		super(name, description, price, out);
		this.rent = rent;
		// this.rent sætter attributen der blev oprettet i klassen lig..
		this.groupNumber = groupNumber;
		// this.groupNumber sætter attributen der blev oprettet i klassen lig..
	}

	@Override // overrider getRent metoden i Ownable klassen
	public int getRent(Player p) {
		// ignore player?
		for (int i = 0; i < Fieldlist.getFields().length; i++)
		// for-loop der kører alle felter igennem.
		{
			if (Fieldlist.getFields()[i] instanceof Plot)
			// Sorterer alle felter der IKKE er plot fra. Vi tjekker altså om
			// det specifik felt er et Plot-felt.
			{
				if (((Plot) Fieldlist.getFields()[i]).getGroupNumber() ==
this.groupNumber
						&& ((Plot) Fieldlist.getFields()[i]).getOwner()
!= this.getOwner())
				// Vi caster felterne til et Plot, så vi kan tilgå
groupNumber,
				// og caster igen så vi kan tilgå Owner af felterne.
				// Og tjekker først om groupNumber er det samme, altså om de
er

				// i samme gruppe. Og derefter om ejeren er forskellig.

				{
					return rent;
					// Returner normal rent.
				}
			}
		}
		return 2 * rent * housecount;
		// Returnerer double rent.
	}


	public int getGroupNumber(){
		return groupNumber;
	}



	public void upgradePlot() {
		if (housecount <= 5) {
			// "if" sørger for at housecount ikke kan blive 5 men ikke mere end
			// 5
			housecount++;
			// øger housecount hver gang upgradePlot metoden kører
		}
	}

	public void downgradePlot() {
		if (!(housecount <= 0)) {

			housecount--;

		}
	}

	public int getHousecount() { // get metode
		return housecount;
	}

	}



package model.fields;

import model.Player;
import view.Out;

public class Tax extends Field
// Tax-klassen arver (extender) fra Field-klassen
{

	private int taxAmmount; // det faste beløb du skal betale
	private int taxRate; // en procent af din formue du skal betale

	public Tax(String name, String description, int taxAmmount, int taxRate, Out out) {
		super(name, description, out);
		this.taxAmmount = taxAmmount;
		// this.taxAmmount sætter attributen der blev oprettet i klassen lig..
		this.taxRate = taxRate;
		// this.taxRate sætter attributen der blev oprettet i klassen lig..
	}

	@Override
	public void landOn(Player p, Out o) {
		if (taxRate > 0)
		// hvis taxRaten (i Fieldlist) er sat større end 0 så kan spilleren
```

```
        // vælge at betale taxAmmount eller taxRate
        {
            if (out.taxAction(taxAmmount))
            // hvis spilleren vælger taxAmmount
            {
                p.getAccount().withdraw(taxAmmount);
                // trækker vha. withdraw metoden
                out.msgGUI("You have paid " + taxAmmount + " in tax");
                // besked i GUI
                System.out.println("You have paid " + taxAmmount + " in tax");
                // besked i konsol
            } else {
                int value = (taxRate * p.getAccount().getSum()) / 100;
                // sætter value lig taxRate (formellen)

                p.getAccount().withdraw(value);
                // trækker value vha. withdraw metoden
                out.msgGUI("You have paid " + value + " in tax");
                // besked i GUI
                System.out.println("You have paid " + value + " in tax");
                // besked i konsol

            }
        } else // ellers (hvis taxRate er 0) så skal spilleren betale taxAmmount
        {
            p.getAccount().withdraw(taxAmmount);
            // trækker vha withdraw metoden
            out.msgGUI("You have paid " + taxAmmount + " in tax");
            // besked i GUI
            System.out.println("You have paid " + taxAmmount + " in tax");
            // besked i konsol
        }
    }

    public int getValue() // get metode
    {
        return taxAmmount;
    }

}
```

BankAccount

```
package model;

public class BankAccount
{
    private int balance;

    public BankAccount(int balance){
        this.balance = balance > 0 ? balance : 0;
    }
```

```
	public int getSum(){
		return balance;
	}

	public void setSum(int value)
	{
		if(value < 0)
		{
			balance = 0;
		}
		else
		{
			balance = value;
		}

	}

	public boolean addSum(int value)
	{
		if (value > 0)
		{
			balance += value;
			return false;
		}

		return true;
	}

	public boolean withdraw(int value)
	{
		if(value > 0 && (balance - value) > 0 )
		{
			balance -= value;
			return true;
		}
		else
		{
			balance = 0;
			return false;
		}
	}
}
```

Dicecup

```
package model;

public class Dicecup
{

	private Die die1 = new Die();
	private Die die2 = new Die();
	public int getSum()
	{
		return die1.getValue() + die2.getValue();
	}

	public Die getDie1()
	{
		return die1;
	}

	public Die getDie2()
	{
		return die2;
	}

	public void roll()
	{
		die1.roll();
		die2.roll();
	}

}
```

Die

```
package model;

public class Die
{

	private int value;

	public void roll()
	{
		value = (int) (Math.random() * 6+1);
	}

	public int getValue()
	{
		return value;
	}

	public void setValue(int value)
	{
		this.value = value;
	}

}
```

Player

```
package model;

import java.util.ArrayList;

import model.cards.FreeJailCard;

public class Player
{

	private String name;
	private BankAccount account;
	private boolean bankruptStatus = false;
	private int playerPos = 0;
	private int breweryCount = 0;
	private int fleetCount = 0;
	private boolean isJailed = false;
	private ArrayList<FreeJailCard> jailcards = new ArrayList<FreeJailCard>();

	public ArrayList<FreeJailCard> getJailcards() {
		return jailcards;
	}

	public void setJailcards(ArrayList<FreeJailCard> jailcards) {
		this.jailcards = jailcards;
	}

	public boolean isJailed() {
		return isJailed;
	}

	public void setJailed(boolean isJailed) {
		this.isJailed = isJailed;
	}

	public Player(String name)
	{
		this.name = name;
		account = new BankAccount(1500);
	}

	public BankAccount getAccount()
	{
		return account;
	}

	public String getName()
	{
		return name;
	}

	public boolean getBankruptStatus()
	{
		return bankruptStatus;
	}


	public void setBankrupt(boolean bankruptStatus)
	{
		this.bankruptStatus = bankruptStatus;
	}

	public int getPlayerPos()
	{
		return playerPos;
	}

	public void setPlayerPos(int newCarPos)
	{
		this.playerPos = newCarPos;
	}

	public void setBreweryCount(int newCount)
	{
		breweryCount = newCount;
	}

	public int getBreweryCount()
	{
		return breweryCount;
	}

	public void setFleetCount(int newCount)
	{
		fleetCount = newCount;
	}

	public int getFleetCount()
	{
		return fleetCount;
	}



}
```

Alltests

```
package tests;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;
```

```
@RunWith(Suite.class)
@SuiteClasses({ ChanceTest.class, JUnitTestBrewery.class, JUnitTestDie.class,
JUnitTestFieldlist.class,
               JUnitTestFleet.class, JUnitTestPlayer.class, JUnitTestStartField.class,
JUnitTestTax.class })
public class AllTests {

}

package tests;

import org.junit.Test;

import controller.Gamecontroller;
import model.Player;
import model.cards.Deck;
import view.FakeOutputTrue;
import view.Out;

public class ChanceTest {

               Out out = new FakeOutputTrue();             // sikre vi kommer uden om GUI
               Gamecontroller gc = new Gamecontroller(out);
               Player p = new Player("Kim");


       @Test
       public void testShuffel() {
               Deck.printCards();
               Deck.shuffleArray();
               Deck.printCards();
               Deck.shuffleArray();
               Deck.printCards();
       }


       @Test
       public void testMove() {
               System.out.println(p.getPlayerPos());

               for (int i = 0; i < 21; i++) {
                       Deck.getCards()[0].doCard(p); //kort 0 rykker to pladser
                       System.out.println(p.getPlayerPos());
               }

       }
}

package tests;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import model.Dicecup;

public class FakeDicecup extends Dicecup {

       private Scanner inputfile;                               // input file
       private String filename = "testdata.txt";                // name of data file

       public FakeDicecup(){
               try {
                       inputfile = new Scanner(new File(filename)); // open file
               }
               catch (IOException e) {                          // error message if the
file cannot be opened
                     System.out.println("Cannot read file " + filename + " in " +
System.getProperty("user.dir"));
         }

               inputfile.close();
       }

       // replaces the method with the same name in Raflebaeger
       @Override
       public int getSum(){
               return TestData.getDice();
       }
}
```

JUnitTestBrewery

```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

import model.Player;
import model.fields.Brewery;
import model.fields.Fieldlist;
import view.FakeOutputTrue;
import view.Out;

public class JUnitTestBrewery
{

        private Out outTrue = new FakeOutputTrue();

        @Test
        public void TC01()
        {
                Player [] p = {new Player("player1"), new Player("player2")};
                new Fieldlist(outTrue);
```

```
		int exRe = p[0].getAccount().getSum() - 150;
		Fieldlist.getFields()[12].landOn(p[0],outTrue);

		int Re = p[0].getAccount().getSum();

		assertEquals(exRe,Re);
	}

	@Test
	public void test2()
	{
		Player [] p = {new Player("p1"),new Player("p2")};
		new Fieldlist(outTrue);

		if(Fieldlist.getFields()[12] instanceof Brewery)
		{
			Brewery br = (Brewery) Fieldlist.getFields()[12];
			br.setOwner(p[0]);
			br.landOn(p[1], outTrue);

			int Re = p[1].getAccount().getSum();
			int exRe = 1500 - (4 * br.getDiceCup().getSum());

			assertEquals(exRe, Re);
		}
	}

}
```

JUnitTestDie

```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

import model.Die;

public class JUnitTestDie {

	@Test
	public void dieTest() {

		Die die1 = new Die();
		int[] terningSide = new int[7];

		for(int i = 0; i <= 10000; i++)
		{
			die1.roll();
			++terningSide[die1.getValue()];
		}

		int test = die1.getValue();
		boolean x = false;

		if(test >= 1 && test <= 6)
		{
			x = true;
		}

		assertEquals(x, true);

		for (int i = 1; i < terningSide.length; i++)
		{
			System.out.println("Count of " + i + " eyes: " + terningSide[i]);
			assertEquals(terningSide[i], 1666, 166);
		}
	}

}
```

JUnitTestFieldlist

```
package tests;
import static org.junit.Assert.*;

import org.junit.Test;

import model.fields.Fieldlist;
import view.FakeOutputTrue;
import view.Out;

public class JUnitTestFieldlist
{


	Out out = new FakeOutputTrue();

	@Test
	public void test01()
	{
		String exsName;
		new Fieldlist(out);

		exsName = Fieldlist.getFields()[1].getName();

		assertEquals(exsName, "Rødovrevej");  // tester at navner passer.
	}
}
```

JUnitTestFleet

```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

import model.Player;
import model.fields.Fieldlist;
import view.FakeOutputTrue;
import view.Out;

public class JUnitTestFleet
{

        private Out out = new FakeOutputTrue();
        private Player bijan = new Player("Bijan");
        private Player kasper = new Player("Kasper");

        @Test
        public void test()
        {
                new Fieldlist(out);

                Fieldlist.getFields()[5].landOn(bijan, out);
                Fieldlist.getFields()[5].landOn(kasper, out);

                assertEquals(1500-200+25, bijan.getAccount().getSum());
                assertEquals(1500-25, kasper.getAccount().getSum());

                Fieldlist.getFields()[15].landOn(bijan, out);
                Fieldlist.getFields()[15].landOn(kasper, out);

                assertEquals(1500-200*2+75, bijan.getAccount().getSum()); // bijan køber
endnu et fleet felt, og kasper lander uheldigvis på samme felt.
                assertEquals(1500-25-50, kasper.getAccount().getSum());   // nu må han
betale 1000, da bijan ejer 2 fleets


                Fieldlist.getFields()[25].landOn(bijan, out);
                Fieldlist.getFields()[25].landOn(kasper, out);

                assertEquals(1500-200*3+175, bijan.getAccount().getSum());// det samme bro,
bare nu med 3 fleets
                assertEquals(1500-25-50-100, kasper.getAccount().getSum());      // betaler
2000

                Fieldlist.getFields()[35].landOn(bijan, out);
                Fieldlist.getFields()[35].landOn(kasper, out);

                assertEquals(1500-200*4+375, bijan.getAccount().getSum());// det samme bro,
bare nu med 3 fleets
```

```
                assertEquals(1500-25-50-100-200, kasper.getAccount().getSum());
        }
}
```

JUnitTestPlayer

```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

import model.Player;

public class JUnitTestPlayer {

        @Test
        public void testPLayer() {

                Player p = new Player("Knud");

                assertEquals("Knud", p.getName());
                assertEquals(1500, p.getAccount().getSum());

                p.getAccount().setSum(0);
                assertEquals(0, p.getAccount().getSum());

                p.getAccount().setSum(100);
                p.getAccount().addSum(200);
                assertEquals(300, p.getAccount().getSum());

                assertEquals(p.getAccount(), p.getAccount());

                p.getAccount().addSum(-500);
                assertEquals(300, p.getAccount().getSum());

                Player p2 = new Player("Brian");

                assertEquals(1500, p2.getAccount().getSum()); // summen må ikke være negativ

                p2.getAccount().setSum(-250000);

                assertEquals(0, p2.getAccount().getSum()); // summen må ikke være negativ

        }

}
```

JUnitTestStartField

```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

import controller.MoveController;
import model.Player;
import model.fields.Fieldlist;
import view.FakeOutputTrue;
import view.Out;

public class JUnitTestStartField
{

	@Test
	public void TC01()
	{
		Player player = new Player("Kasper");
		Out out = new FakeOutputTrue();
		MoveController moveCon = new MoveController();
		new Fieldlist(out);

		player.setPlayerPos(35);
		moveCon.movePlayer(5, player);

		Fieldlist.getFields()[player.getPlayerPos()].landOn(player, out);

		int exRes = 1500 + 200;
		int res = player.getAccount().getSum();

		assertEquals(res, exRes);
	}

}
```

JUnitTestTax

```
package tests;

import static org.junit.Assert.*;

import org.junit.Test;

import model.fields.*;
import view.FakeOutputFalse;
import view.FakeOutputTrue;
import view.Out;
import model.Player;;
```

```
public class JUnitTestTax
{
	Out outTrue = new FakeOutputTrue();
	Out outFalse = new FakeOutputFalse();

	@Test
	public void TC01()
	{

		Player p = new Player("Kasper");

		int exRes = (p.getAccount().getSum() - (10 * 1500) / 100);

		new Fieldlist(outFalse);
		Fieldlist.getFields()[4].landOn(p, outFalse);

		int res = p.getAccount().getSum();

		assertEquals(exRes, res);
	}

	@Test
	public void TC02()
	{
		Player p = new Player("Kasper");

		int exRes = p.getAccount().getSum() - 100;

		new Fieldlist(outTrue);
		Fieldlist.getFields()[38].landOn(p, outTrue);

		int res = p.getAccount().getSum();

		assertEquals(exRes, res);
	}
}
```

TestData

```
package tests;

/**
 * Testdata supplies test data to other classes
 * Version 2.00
 * Skrevet af: Agner Fog
 * Dato: 2016-10-23
 * The method getDice returns predefined values from file "testdata.txt" instead of random
numbers.
 * The method getUserDecision returns a number from the file testdata.txt to simulate a
user decision.
 */
```

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;

import desktop_resources.GUI;

// Use this class to supply testdata to other classes. Everything is static.
public class TestData {

       static private Scanner inputfile = null;            // input file
       static private String filename = "testdata.txt";    // name of test data input file
       static private int[] linedata = new int[2];        // data extracted from one line
of input file

       static protected Scanner getScanner () {
              if (inputfile == null) {
                     try {
                            inputfile = new Scanner(new File(filename)); // open file
                     }
                     catch (IOException e) { // error message if the file cannot be opened
                   System.out.println("Cannot read file " + filename + " in " +
System.getProperty("user.dir"));
                   GUI.close();
                   System.exit(0);
               }
              }
              return inputfile;
       }

       private static void readNextLine ()
       {
        if (getScanner().hasNextLine())
        {
            String line = inputfile.nextLine();  // read one line
            String[] fields = line.split(",");   // split the comma-separated line into
fields

            for (int i = 0; i < fields.length; i++)
            {
              int value = Integer.parseInt(fields[i]); // converert each field to an
integer
              linedata[i] = value;
            }
        }
        else { // there are no more lines in the file
                     System.out.println("End of file " + filename);
                     inputfile.close();
                     System.exit(0);
              }
       }

       static public int getDice() {
              readNextLine();
              int x = linedata[0] + linedata[1];
              return x;
```

```
	}

    public static int[] getLinedata() {
		return linedata;
	}


}
```

FakeOutputFalse

```
package view;

import model.Player;

public class FakeOutputFalse extends Out {

	@Override
	public void winnerPrint(Player p) {
		// TODO Auto-generated method stub

	}

	@Override
	public void oneMoreGame() {
		// TODO Auto-generated method stub

	}

	@Override
	public int howManyPlayers() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public void drawGameboard() {
		// TODO Auto-generated method stub

	}

	@Override
	public boolean taxAction(int price) {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
	public void setGUIDice(int die1, int die2) {
		// TODO Auto-generated method stub
	}

	@Override
	public void setGUIBalance(Player p) {
		// TODO Auto-generated method stub

	}

	@Override
	public void setcar(Player p) {
		// TODO Auto-generated method stub

	}

	@Override
	public void msgGUI(String s) {
		// TODO Auto-generated method stub

	}

	@Override
	public void addPlayersToGUI(Player player) {
		// TODO Auto-generated method stub

	}

	@Override
	public boolean setTestMode() {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
	public void verificationOfPurchase() {
		// TODO Auto-generated method stub

	}

	@Override
	public void deniedPurchase() {
		// TODO Auto-generated method stub

	}

	@Override
	public void rollDiceText() {
		// TODO Auto-generated method stub

	}

	@Override
	public boolean shopField(int price, Player p) {
		// TODO Auto-generated method stub
		return false;
```

```
}

@Override
public void setColor(Player p) {
	// TODO Auto-generated method stub

}

@Override
public void payedRent(Player p, int rent) {
	// TODO Auto-generated method stub

}

@Override
public void landMSG(Player p) {
	// TODO Auto-generated method stub

}

@Override
public void removeOwner(int fieldNum) {
	// TODO Auto-generated method stub

}

@Override
public String Jailaction(Player p, String[] array) {
	// TODO Auto-generated method stub
	return null;
}

@Override
public void removeCar(Player p) {
	// TODO Auto-generated method stub

}

@Override
public void CardsOut(String dis) {
	// TODO Auto-generated method stub

}

@Override
public void closeGame() {
	// TODO Auto-generated method stub

}

@Override
public void build(Player p, Out out, String farve) {
	// TODO Auto-generated method stub
	}

	@Override
	public String whereToBuild(String[] fArray) {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public void BuildHouse(int index, int houseCount) {
		// TODO Auto-generated method stub

	}

	@Override
	public void BuildHotel(int index, boolean hasHotel) {
		// TODO Auto-generated method stub

	}

	@Override
	public boolean sellOrBuy() {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
	public String whereToSell(String[] fArray) {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public boolean shopOrNot() {
		// TODO Auto-generated method stub
		return false;
	}


}
```

FakeOutputTrue

```
package view;

import model.Player;

public class FakeOutputTrue extends Out {

	@Override
	public void winnerPrint(Player p) {
		// TODO Auto-generated method stub
```

```
    }

    @Override
    public void oneMoreGame() {
            // TODO Auto-generated method stub

    }

    @Override
    public int howManyPlayers() {
            // TODO Auto-generated method stub
            return 3;
    }

    @Override
    public void drawGameboard() {
            // TODO Auto-generated method stub

    }

    @Override
    public boolean taxAction(int price) {
            // TODO Auto-generated method stub
            return true;
    }

    @Override
    public void setGUIDice(int die1, int die2) {
            // TODO Auto-generated method stub

    }

    @Override
    public void setGUIBalance(Player p) {
            // TODO Auto-generated method stub

    }

    @Override
    public void setcar(Player p) {
            // TODO Auto-generated method stub

    }

    @Override
    public void msgGUI(String s) {
            // TODO Auto-generated method stub

    }

    @Override
    public void addPlayersToGUI(Player player) {
            // TODO Auto-generated method stub
```

```
}

@Override
public boolean setTestMode() {
        // TODO Auto-generated method stub
        return true;
}

@Override
public void verificationOfPurchase() {
        // TODO Auto-generated method stub

}

@Override
public void deniedPurchase() {
        // TODO Auto-generated method stub

}

@Override
public void rollDiceText() {
        // TODO Auto-generated method stub

}

@Override
public boolean shopField(int price, Player p) {
        // TODO Auto-generated method stub
        return true;
}

@Override
public void setColor(Player p) {
        // TODO Auto-generated method stub

}

@Override
public void payedRent(Player p, int rent) {
        // TODO Auto-generated method stub

}

@Override
public void landMSG(Player p) {
        // TODO Auto-generated method stub

}

@Override
public void removeOwner(int fieldNum) {
        // TODO Auto-generated method stub
```

```
}

@Override
public String Jailaction(Player p, String[] array) {
	// TODO Auto-generated method stub
	return null;
}

@Override
public void removeCar(Player p) {
	// TODO Auto-generated method stub

}

@Override
public void CardsOut(String dis) {
	// TODO Auto-generated method stub

}

@Override
public void closeGame() {
	// TODO Auto-generated method stub

}

@Override
public void build(Player p, Out out, String farve) {
	// TODO Auto-generated method stub

}

@Override
public String whereToBuild(String[] fArray) {
	// TODO Auto-generated method stub
	return null;
}

@Override
public void BuildHouse(int index, int houseCount) {
	// TODO Auto-generated method stub

}

@Override
public void BuildHotel(int index, boolean hasHotel) {
	// TODO Auto-generated method stub

}

@Override
public boolean sellOrBuy() {
	// TODO Auto-generated method stub
	return false;
	}

	@Override
	public String whereToSell(String[] fArray) {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public boolean shopOrNot() {
		// TODO Auto-generated method stub
		return false;
	}


}
```

Language

```
package view;

import model.Player;

/*
 * Class wrote by: Kasper Leiszner
 */

public class Language
{
	private static String[] fieldName =
		{
				"Start",				// felt 0
				"Rødovrevej",
				"Prøv lykken",
				"Hvidovrevej",
				"Skat",
				"Øresund",
				"Roskildevej",
				"Prøv lykken",
				"Valby Langgade",
				"Allegade",
				"Fængsel",				// felt 10
				"Frederiksberg Allé",
				"Tuborg",
				"Bulowsvej",
				"Gammel Kongevej",
				"D.F.D.S",
				"Bernstorffsvej",
				"Prøv lykken",
				"Hellerupvej",
				"Strandvejen",
				"Helle",				//felt 20
				"Trianglen",
                              "Prøv lykken",
                              "Østerbrogade",
                              "Grønningen",
                              "Ø.S.",
                              "Bredgade",
                              "Kgs. Nytorv",
                              "Calsberg",
                              "Østergade",
                              "Gå i fængsel",                     //felt 30
                              "Amagertorv",
                              "Vimmelskaftet",
                              "Prøv lykken",
                              "Nygade",
                              "Bornholm",
                              "Prøv lykken",
                              "Frederiksberggade",
                              "Skat",
                              "Rådhuspladsen"                     // felt 39
               };

       public static String[] getCardDescriptions() {
               return cardDescriptions;
       }

       private static String[] cardDescriptions =
               {
                              "Ejendomsskatterne er steget, ekstraudgifterne er: kr. 50,00
pr. hus, kr. 125,00 pr. hotel.",
                              "Ryk frem til Frederiksberg Allé. Hvis De passerer START,
indkassér da kr. 200,00",
                              "De har lagt penge ud til sammenskudsgilde. Mærkværdigvis
betaler alle straks. Modtag fra hver medspiller kr. 25,00",
                              "Ryk tre felter tilbage",
                              "De modtager Deres aktieudbytte. Modtag kr. 50 af banken.",
                              "Grundet dyrtiden har De fået gageforhøjelse. Modtag kr.
25,00.",
                              "Betal for vognvask og smøring kr. 10,00.",
                              "Deres præmieobligation er udtrukket. De modtager kr. 100 af
banken.",
                              "Deres præmieobligation er udtrukket. De modtager kr. 100 af
banken.",
                              "Ryk frem til START.",
                              "Ryk frem til START.",
                              "De skal holde familiefest og fået et tilskud fra hver
medspiller på kr. 25,00.",
                              "Tag ind på Rådhuspladsen.",
                              "Tag med Øresundsredderiet. Flyt brikken frem, og hvis De
passerer "START", indkassér da kr. 200,00.",
                              "De har købt 4 nye dæk til Deres vogn. Betal kr. 50,00.",
                              "De har fået en parkeringsbøde. Betal kr. 20,00 i bøde.",
                              "Betal kr. 10 for levering af 2 kasser øl.",
                              "Tag med den nærmeste færge. Flyt brikken frem, og hvis De
passerer "START", indkassér da kr. 200.",
                              "Gå i fængsel. Ryk direkte til fængslet. Selv om de passerer
"START", indkasserer de ikke kr. 200.",
                              "Gå i fængsel. Ryk direkte til fængslet. Selv om de passerer
"START", indkasserer de ikke kr. 200.",
                              "Oliepriserne er steget, og De skal betale: kr. 25 pr. Hus, 
kr. 100 pr. hotel",
                              "Modtag udbytte af Deres aktier: kr. 50,00.",
                              "Modtag udbytte af Deres aktier: kr. 50,00.",
                              "Ryk brikken frem til det nærmeste rederi og betal lejeren to 
gange leje han ellers er berettiget til. Hvis selskabet ikke ejes af nogen kan du købe det 
af banken.",
                              "Ryk frem til Strandvejen. Hvis du passerer start, indkassér 
kr. 200,00.",
                              "Betal deres bilforsikring - kr. 50,00.",
                              "De havde en række med elleve rigtige i tipning. Modtag kr. 
50.",
                              "De har vundet i klasselotteriet. Modtag kr. 25.",
                              "De har vundet i klasselotteriet. Modtag kr. 25.",
                              "Ryk frem til Vimmelskaftet. Hvis de passerer start, indkassér 
da kr. 200,00.",
                              "Ryk tre felter tilbage",
                              "Ryk tre felter tilbage",
                              "I anledning af kongens fødselsdag benådes de herved for 
fængsel. Dette kort kan opbevares indtil De får brug for det eller De kan sælge det.",
                              "I anledning af kongens fødselsdag benådes de herved for 
fængsel. Dette kort kan opbevares indtil De får brug for det eller De kan sælge det.",
                              "Betal kr. 150,00. for reparation af Deres vogn.",
                              "De har solgt nogle gamle møbler på auktion. Modtag kr. 50 af 
banken.",
                              "De har været en tur i udlandet og haft mange cigaretter med 
hjem. Betal told kr. 10.",
                              "Ryk frem til Grønningen. Hvis de passerer start indkassér da 
kr. 200,00.",
                              "Kommunen har eftergivet et kvartals skat. Hæv i banken kr. 
150,00.",
                              "De har lagt penge ud til et sammenskudsgilde. Mærkværdigvis 
betaler alle straks. Modtag fra hver medspiller kr. 25,00.",
                              "De har kørt frem for "Fuldt stop". Betal kr. 50,00 i bøde.",
                              "De har modtaget deres tandlægeregning. Betal kr. 10,00.",
                              "Værdien af egen avl fra nyttehaven udgør kr. 100,00 som De 
modtager af banken.",



               };

        private static String[] yesNo =
               {
                              "Ja",
                              "Nej"
               };

        private static String[] fieldDescription =
               {
                              "Start", //[0]
                              "Grund", //[1]
                              "Parkering", //[2]
                              "Bryggeri", //[3]
                              "SKAT", //[4]
                              "Redderi", //[5]
```

```
				"Fængsel", //[6]
				"Ryk i fængsel" //[7]
		};

	private static String testString[] =
		{
				"Test mode options",
				"Run Test Mode",
				"Run Normal Game"
		};

	private static String winnerText = "YOYOYOYO! Vinderen af matador-master-badboy-
game-2017 er:";
	private static String endText = "Vil du spille igen?";
	private static String playerCountText = "Hvor mange spilere er i?";
	private static String nameOfPlayer = "Navnet på spiller";
	private static String rollDice = "Kast terningerne";
	private static String makeYourChoice = "Tag dit valg";
	private static String purchase = "Købet gik igennem";
	private static String deniedPurchase = "Købet blev afvist";
	private static String pay = "Betal";
	private static String tenPercent = "eller betal 10% af din balance";
	private static String willYouBuy = "Vil du købe denne grund?";
	private static String payedRent = "Feltet er ejet, du betalte leje til ejeren";
	private static String ownField = "Dette er dit eget felt";

	public static String getPayedRent()
	{
		return payedRent;
	}

	public static String ownField()
	{
		return ownField;
	}

	public static void setPayedRent(String payedRent)
	{
		Language.payedRent = payedRent;
	}

	public static String[] getFieldDecription()
	{
		return fieldDescription;
	}

	public static String[] getYesNo()
	{
		return yesNo;
	}

	public static String getWillYouBuy()
	{
		return willYouBuy;
	}
```

```
public static String getTenPercent()
{
	return tenPercent;
}

public static String getPayMSG()
{
	return pay;
}

public static String getMakeYourChoiceMSG()
{
	return makeYourChoice;
}

public static String getPlayerCountText()
{
	return playerCountText;
}

public static String getNameOfPlayer()
{
	return nameOfPlayer;
}

public static String getEndText()
{
	return endText;
}

public static String getWinnerText()
{
	return winnerText;
}

public static String[] getFieldNames()
{
	return fieldName;
}

public static String rollDiceText()
{
	return rollDice;
}

public static String testModeStrings(int x)
{
	return testString[x];
}

public static String purchaseString()
{
	return purchase;
}
```

```
	public static String deniedPurchaseString()
	{
		return deniedPurchase;
	}

	public static String payedRent()
	{
		return payedRent;
	}

	public static String whoseTurn(Player p)
	{
		return "Det er " + p.getName() + "'s tur";
	}

	public static String extraTurn(Player p)
	{
		return p.getName() + " slog to ens, og belønnes med en ekstra tur";
	}
}
package view;

import model.Player;

public abstract class Out {

	public abstract void winnerPrint(Player p);

	public abstract void oneMoreGame();

	public abstract int howManyPlayers();

	public abstract void drawGameboard();

	public abstract boolean taxAction(int price);

	public abstract void setGUIDice(int die1 , int die2);

	public abstract void setGUIBalance(Player p);

	public abstract void setcar(Player p);

	public abstract void msgGUI(String s);

	public abstract void addPlayersToGUI(Player player);

	public abstract boolean setTestMode();

	public abstract void verificationOfPurchase();

	public abstract void deniedPurchase();
```

```java
	public abstract void rollDiceText();

	public abstract boolean shopField(int price, Player p);

	public abstract void setColor(Player p);

	public abstract void payedRent(Player p, int rent);

	public abstract void landMSG(Player p);

	public abstract void removeOwner(int fieldNum);

	public abstract String Jailaction(Player p, String[] array);

	public abstract void removeCar(Player p);

	public abstract void CardsOut(String dis);

	public abstract void closeGame();

	public abstract void build(Player p, Out out, String farve);

	public abstract String whereToBuild(String[] fArray);

	public abstract void BuildHouse(int index, int houseCount);

	public abstract void BuildHotel(int index, boolean hasHotel);

	public abstract boolean sellOrBuy();

	public abstract String whereToSell(String[] fArray);

	public abstract boolean shopOrNot();

}
```

Output

```java
package view;

/*
 * Class wrote by: Troels Lund and Kasper Leiszner
 */

import desktop_fields.Street;
import desktop_resources.GUI;
import model.Player;
import model.fields.Fieldlist;

public class Output extends Out
```

```
{

	public void winnerPrint(Player p)
	{
		System.out.println(Language.getWinnerText() + " " + p.getName());
		GUI.showMessage(Language.getWinnerText() + " " + p.getName());
	}

	public void oneMoreGame()
	{
		System.out.println(Language.getEndText());
		GUI.showMessage(Language.getEndText());
	}

	public int howManyPlayers()
	{
		int result = 
Integer.parseInt(GUI.getUserSelection(Language.getPlayerCountText(), "2","3","4","5","6"));
		System.out.println(result + " players are made");
		return result;
	}

	public void drawGameboard()
	{
		model.fields.Field[] logicField = Fieldlist.getFields();
		desktop_fields.Field[] guiField = new 
desktop_fields.Field[logicField.length];

		for (int i = 0; i < logicField.length; i++)
		{
			guiField[i] = new Street.Builder()
					.setTitle(logicField[i].getName())
					.setDescription(logicField[i].getDescription())
					.setRent("" + logicField[i].getValue())
					.build();
		}

		GUI.create(guiField);
	}

	public boolean taxAction(int price)
	{
		Boolean result = 
GUI.getUserLeftButtonPressed(Language.getMakeYourChoiceMSG(), Language.getPayMSG() + " " + 
price , Language.getTenPercent());
		System.out.println("You did pay the normal tax " + result + " .If false you 
payed 10%");
		return result;
	}

	public void setGUIDice(int die1 , int die2)
	{
		System.out.println("You rolled " + die1 + " " + die2 + " The sum is " + 
(die1 + die2));
		GUI.setDice(die1,die2);
	}
	public void setGUIBalance(Player p)
	{
		System.out.println("Your new balance is " + p.getAccount().getSum());
		GUI.setBalance(p.getName(), p.getAccount().getSum());
	}

	public void setcar(Player p)
	{
		System.out.println(p.getName() + " - Gui car is moved to field number " +
(p.getPlayerPos()+1));

		GUI.setCar(p.getPlayerPos()+1,p.getName());
	}

	public void msgGUI(String s)
	{
		System.out.println(s);
		GUI.showMessage(s);
	}

	public void rollDiceText()
	{
		GUI.getUserButtonPressed("", Language.rollDiceText());
	}

	public void addPlayersToGUI(Player player)
	{
		GUI.addPlayer(player.getName(), player.getAccount().getSum());
		GUI.setCar(1, player.getName());
	}

	public boolean shopField(int price, Player p)
	{
		boolean x = GUI.getUserLeftButtonPressed(p.getName() + " " +
Language.getWillYouBuy() + " " + price, Language.getYesNo()[0], Language.getYesNo()[1]);


		return x;
	}

	public boolean setTestMode()
	{
		return
GUI.getUserLeftButtonPressed(Language.testModeStrings(0),Language.testModeStrings(1),Langua
ge.testModeStrings(2));
	}

	public void verificationOfPurchase()
	{
		GUI.showMessage(Language.purchaseString());
		System.out.println("You bought this field");
	}

	public void deniedPurchase()
	{
		GUI.showMessage(Language.deniedPurchaseString());
		System.out.println("You didn't buy the field");
	}

	public void removeCar(Player p)
	{
		GUI.removeCar(p.getPlayerPos()+1, p.getName());
		System.out.println(p.getName() + " car got removed in GUI");
	}

	public void setColor(Player p)
	{
		GUI.setOwner(p.getPlayerPos()+1, p.getName());
		System.out.println("Owner was set");
	}

	public void payedRent(Player p, int rent)
	{
		GUI.showMessage(p.getName() + ", " + Language.payedRent() + " " + rent );
		System.out.println("Field is owned, you paid the rent");
	}

	public void ownField()
	{
		GUI.showMessage("It's your own field - nothing happened");
		System.out.println("It's your own field - nothing happened");
	}

	public void landMSG(Player p)
	{
		GUI.showMessage(p.getName() + " has landed on " + 
Fieldlist.getFields()[p.getPlayerPos()].getClass().getSimpleName());
		System.out.println(p.getName() + " has landed on " + 
Fieldlist.getFields()[p.getPlayerPos()].getClass().getSimpleName());
	}

	public String Jailaction(Player p, String[] array){
		return GUI.getUserButtonPressed("Vil du slå med terningerne eller betale", 
array);
	}

	@Override
	public void removeOwner(int fieldNum)
	{
		//TODO
		GUI.removeOwner(fieldNum);
	}

	public void CardsOut(String dis){
		GUI.displayChanceCard("Chancekort: " + dis );
	}
	public void closeGame()
	{
		GUI.close();
	}
	public void build(Player p, Out out, String farve){

	}

	public String whereToBuild(String[] fArray){
		String result = GUI.getUserSelection("Hvor vil du bygge? - Det vil koste dig 
200kr at bygge", fArray);
		return result;
	}

	public String whereToSell(String[] fArray){
		String result = GUI.getUserSelection("Hvad for en bygning vil du sælge?", 
fArray);
		return result;
	}

	public void BuildHouse(int index, int houseCount){
		GUI.setHouses(index, houseCount);
	}

	public void BuildHotel(int index, boolean hasHotel){
		GUI.setHotel(index, hasHotel);

	}

	public boolean sellOrBuy(){
		return GUI.getUserLeftButtonPressed("Vil du købe eller sælg husse?", "køb", 
"sælg");

	}

	public boolean shopOrNot(){
		return GUI.getUserLeftButtonPressed("vil du handle eller ej?", "ja", "nej");

	}

	public String convertToColor(int group){
		switch(group){
		case 0: return "lyseblå";
		case 1: return "lyserød";
		case 2: return "grøn";
		case 3: return "grå";
		case 4: return "rød";
		case 5: return "hvid";
		case 6: return "gul";
		case 7: return "brun";
		}
		return null;

	}

}
```