```java
package run;
import control.GameController;
public class Launch {
    public static void main(String[] args) {
        GameController game = new GameController(1);
        game.init();
    }

}

```

```java
package run;
import control.GameController;
public class Launch_Showcase {
    public static void main(String[] args) {
        GameController game = new GameController(2);
        game.init();
    }

}

```

```java
package boundary;

import java.awt.Color;

import javax.swing.JOptionPane;

import desktop_codebehind.Car;
import desktop_fields.Brewery;
import desktop_fields.Chance;
import desktop_fields.Field;
import desktop_fields.Jail;
import desktop_fields.Refuge;
import desktop_fields.Shipping;
import desktop_fields.Start;
import desktop_fields.Street;
import desktop_fields.Tax;
import desktop_resources.GUI;
import entity.Player;

public class GUIController {
    //Game methods
    public void createField() {
        // Array that holds the fields for the GUI
        Field[] field = new Field[40];
        field[0] = new Start.Builder().setTitle("Start")
                .setDescription("Start").setSubText("Start")
                .setBgColor(Color.RED).build();
        field[1] = new Street.Builder().setTitle("Rødovrevej")
                .setDescription("Rødovrevej").setBgColor(Color.BLUE)
                .setSubText("Price: 1200").setRent("Rent: 1000").build();
        field[2] = new Chance.Builder().build();
        field[3] = new Street.Builder().setTitle("Hvidovrevej")
                .setDescription("Hvidovrevej").setBgColor(Color.BLUE)
                .setSubText("Price: 1200").setRent("Rent: 1000").build();
        field[4] = new Tax.Builder().setTitle("Statskat")
                .setDescription("Pay: 4000 or 10%").setBgColor(Color.ORANGE)
                .setSubText("Pay: 4000 or 10%").build();
        field[5] = new Shipping.Builder().setTitle("Helsingør-Helsingborg")
                .setDescription("Helsingør-Helsingborg").setBgColor(Color.GRAY)
                .setSubText("Price: 4000").setRent("Rent: 500").build();
        field[6] = new Street.Builder().setTitle("Roskildevej")
                .setDescription("Roskildevej").setBgColor(Color.PINK)
                .setSubText("Price: 2000").setRent("Rent: 1000").build();
        field[7] = new Chance.Builder().build();
        field[8] = new Street.Builder().setTitle("Valby Langgade")
                .setDescription("Valby Langgade").setBgColor(Color.PINK)
                .setSubText("Price: 2000").setRent("Rent: 1000").build();
        field[9] = new Street.Builder().setTitle("Allégade")
                .setDescription("Allégade").setBgColor(Color.PINK)
                .setSubText("Price: 2400").setRent("Rent: 1000").build();
        field[10] = new Jail.Builder().setTitle("Fængsel").setDescription("Fængsel")
                .setSubText("Fængsel").build();
        field[11] = new Street.Builder().setTitle("Fredriksberg Allé")
                .setDescription("Fredriksberg Allé").setBgColor(Color.GREEN)
                .setSubText("Price: 2800").setRent("Rent: 2000").build();
        field[12] = new Brewery.Builder().setTitle("Tuborg")
                .setDescription("Tuborg").setBgColor(Color.CYAN)
                .setSubText("Price: 3000").setRent("80/200 x dice").build();
        field[13] = new Street.Builder().setTitle("Bulowsvej")
                .setDescription("Bulowsvej").setBgColor(Color.GREEN)
                .setSubText("Price: 2800").setRent("Rent: 2000").build();
        field[14] = new Street.Builder().setTitle("Gl Kongevej")
```

```
63                  .setDescription("Gl Kongevej").setBgColor(Color.GREEN)
64                  .setSubText("Price: 3200").setRent("Rent: 2000").build();
65          field[15] = new Shipping.Builder().setTitle("Mols-Linien")
66                  .setDescription("Mols-Linien").setBgColor(Color.CYAN)
67                  .setSubText("Price: 4000").setRent("Rent: 500").build();
68          field[16] = new Street.Builder().setTitle("Bernstorffsvej")
69                  .setDescription("Bernstorffsvej").setBgColor(Color.GRAY)
70                  .setSubText("Price: 3600").setRent("Rent: 2000").build();
71          field[17] = new Chance.Builder().build();
72          field[18] = new Street.Builder().setTitle("Hellerupvej")
73                  .setDescription("Hellerupvej").setBgColor(Color.GRAY)
74                  .setSubText("Price: 3600").setRent("Rent: 2000").build();
75          field[19] = new Street.Builder().setTitle("Strandvejen")
76                  .setDescription("Strandvejen").setBgColor(Color.GRAY)
77                  .setSubText("Price: 4000").setRent("Rent: 2000").build();
78          field[20] = new Refuge.Builder().setTitle("Helle")
79                  .setDescription("Price 5000").setBgColor(Color.ORANGE)
80                  .setSubText("Parkering").build();
81          field[21] = new Street.Builder().setTitle("Trianglen")
82                  .setDescription("Trianglen").setBgColor(Color.RED)
83                  .setSubText("Price: 4400").setRent("Rent: 3000").build();
84          field[22] = new Chance.Builder().build();
85          field[23]= new Street.Builder().setTitle("Østerbrogade")
86                  .setDescription("Østerbrogade").setBgColor(Color.RED)
87                  .setSubText("Price: 4400").setRent("Rent: 3000").build();
88          field[24] = new Street.Builder().setTitle("Grønningen")
89                  .setDescription("Grønningen").setBgColor(Color.RED)
90                  .setSubText("Price: 4800").setRent("Rent: 3000").build();
91          field[25] = new Shipping.Builder().setTitle("Gedser-Rostock")
92                  .setDescription("Gedser-Rostock").setBgColor(Color.CYAN)
93                  .setSubText("Price: 4000").setRent("Rent: 500").build();
94          field[26] = new Street.Builder().setTitle("Bredgade")
95                  .setDescription("Bredgade").setBgColor(Color.WHITE)
96                  .setSubText("Price: 5200").setRent("Rent: 3000").build();
97          field[27] = new Street.Builder().setTitle("Kgs Nytorv")
98                  .setDescription("Kgs Nytorv").setBgColor(Color.WHITE)
99                  .setSubText("Price: 5200").setRent("Rent: 3000").build();
100         field[28] = new Brewery.Builder().setTitle("Coca-Cola")
101                 .setDescription("Coca-Cola").setBgColor(Color.CYAN)
102                 .setSubText("Price: 3000").setRent("80/200 x dice").build();
103         field[29] = new Street.Builder().setTitle("Østergade")
104                 .setDescription("Østergade").setBgColor(Color.WHITE)
105                 .setSubText("Price: 5600").setRent("Rent: 3000").build();
106         field[30] = new Jail.Builder().setDescription("Ryk til Fængsel")
107                 .setSubText("Ryk Direkte I Fængsel").build();
108         field[31] = new Street.Builder().setTitle("Amagertorv")
109                 .setDescription("Amagertorv").setBgColor(Color.YELLOW)
110                 .setSubText("Price: 6000").setRent("Rent: 4000").build();
111         field[32] = new Street.Builder().setTitle("Vimmelskaftet")
112                 .setDescription("Vimmelskaftet").setBgColor(Color.YELLOW)
113                 .setSubText("Price: 6000").setRent("Rent: 4000").build();
114         field[33] = new Chance.Builder().build();
115         field[34] = new Street.Builder().setTitle("Nygade")
116                 .setDescription("Nygade").setBgColor(Color.YELLOW)
117                 .setSubText("Price: 6400").setRent("Rent: 4000").build();
118         field[35] = new Shipping.Builder().setTitle("Rødby-Puttgarden")
119                 .setDescription("Rødby-Puttgarden").setBgColor(Color.CYAN)
120                 .setSubText("Price: 4000").setRent("Rent: 500").build();
121         field[36] = new Chance.Builder().build();
122         field[37] = new Street.Builder().setTitle("Frederiksberggade")
123                 .setDescription("Frederiksberggade").setBgColor(Color.MAGENTA)
124                 .setSubText("Price: 7000").setRent("Rent: 4000").build();
```

```
125         field[38] = new Tax.Builder().setTitle("Indkomstskat")
126                 .setDescription("Pay: 2000").setBgColor(Color.ORANGE)
127                 .setSubText("Pay: 2000").build();
128         field[39] = new Street.Builder().setTitle("Rådhuspladsen")
129                 .setDescription("Rådhuspladsen").setBgColor(Color.MAGENTA)
130                 .setSubText("Price: 8000").setRent("Rent: 4000").build();
131 
132         GUI.create(field);
133     }
134 
135     public int playerAmount() {
136         return Integer.parseInt(JOptionPane.showInputDialog("Vælg antallet af spillere
    (mellem 3-6)"));
137     }
138     public int playerAmountshowcase() {
139         return Integer.parseInt(JOptionPane.showInputDialog("Vælg antallet af spillere
    (mellem 2-6) \n DETTE ER EN DEMO KØRSEL AF SPILLET"));
140     }
141 
142     public void playerAmountError(){
143         JOptionPane.showMessageDialog(null, "Du skal vælge mellem 3-6 spiller");
144     }
145     public void showMessage(String message) {
146         GUI.showMessage(message);
147     }
148 
149     public void createPlayers(int playerAmount, Player[] player) {
150         // Array that holds the cars for the GUI
151         Car[] car = new Car[6];
152         car[0] = new Car.Builder().primaryColor(Color.blue).build();
153         car[1] = new Car.Builder().primaryColor(Color.red).build();
154         car[2] = new Car.Builder().primaryColor(Color.yellow).build();
155         car[3] = new Car.Builder().primaryColor(Color.green).build();
156         car[4] = new Car.Builder().primaryColor(Color.white).build();
157         car[5] = new Car.Builder().primaryColor(Color.black).build();
158 
159         for (int i = 0; i < playerAmount; i++) {
160             player[i] = new Player(JOptionPane.showInputDialog("Skriv navnet for spiller
    "+(i + 1) + ":"));
161             GUI.addPlayer(player[i].getName(), player[i].account.getScore(),
162                     car[i]);
163             GUI.setCar(player[i].getPosition() + 1, player[i].getName());
164         }
165     }
166 
167     public void nextPlayer(Player[] player, int currentPlayer) {
168         GUI.showMessage(player[currentPlayer].getName()+"'s tur til at slå.");
169     }
170 
171     public void showDice(int dice1, int dice2) {
172         GUI.setDice(dice1, dice2);
173     }
174 
175     public void setOwner(Player player) {
176         GUI.setOwner(player.getPosition()+1, player.getName());
177     }
178 
179     public void updatePosition(Player[] player, int currentPlayer, int diceSum) {
180         // Remove car from old position on board
181         GUI.removeAllCars(player[currentPlayer].getName());;
182         player[currentPlayer].movePosition(diceSum);
183         // Sets car on new position on board
```

```
184         GUI.setCar(player[currentPlayer].getPosition() + 1,
185                 player[currentPlayer].getName());
186     }
187     public void removeCar(Player player) {
188         GUI.removeAllCars(player.getName());
189     }
190     public void setCar(Player player) {
191         GUI.setCar(player.getPosition(), player.getName());
192     }
193 
194     public void newPosition(Player player) {
195         GUI.removeAllCars(player.getName());
196         GUI.setCar(player.getPosition()+1, player.getName());
197     }
198     public void removePlayer(Player[] player, int currentPlayer) {
199         GUI.removeCar(player[currentPlayer].getPosition() + 1,
    player[currentPlayer].getName());
200         GUI.showMessage("Du er fallit.");
201     }
202 
203     public void showWin(Player[] player, int playerAmount) {
204         for (int i = 0; i < playerAmount; i++) {
205             if (!player[i].getStatus()) {
206                 GUI.showMessage(player[i].getName() + " har vundet!");
207             }
208         }
209         GUI.close();
210     }
211 
212     //Field methods
213     public boolean buyField(String name, int price) {
214         return GUI.getUserLeftButtonPressed(name +
215                 " har ingen ejer, vil du købe grunden for " + price + "?", "Ja",
216                 "Nej");
217     }
218 
219     public void fieldBought(String name) {
220         GUI.showMessage("Du har købt " + name);
221     }
222 
223     public void pastStart(){
224         GUI.showMessage("Du har passeret start");
225     }
226 
227     public void afterJail(){
228         GUI.showMessage("Du forlod fængslet, og rykker summen på terningerne");
229     }
230     public void moveToJail() {
231         GUI.showMessage("Du har stødt på en politiracia, du er blevet arresteret, og bliver
    forflyttet til fængslet");
232     }
233 
234     public void threePair(){
235         GUI.showMessage("Du har slået 3 par i træk, du ryger i fængsel");
236     }
237 
238     public void twoPair(){
239         GUI.showMessage("Tryk for at slå igen, hvis du slår to ens igen ryger du i
    fængsel");
240     }
241 
242     public void onePair(){
```

```
243         GUI.showMessage("Tryk for at slå igen");
244     }
245
246     public void fieldRefused(String name) {
247         GUI.showMessage("Du har ikke købt " + name);
248     }
249
250     public void fieldRefusedPrice(String name) {
251         GUI.showMessage("Du har ikke nok penge til at købe " + name);
252     }
253
254     public void fieldOwnedByPlayer(String name) {
255         GUI.showMessage(name
256                 + " er ejet af dig selv og der sker derfor ikke noget..");
257     }
258
259     public void fieldTax(String fieldName, String playerName, int price) {
260         GUI.showMessage(fieldName + " er ejet af " + playerName
261                 + ", " + price
262                 + " kr vil blive flyttet til "
263                 + playerName + "'s konto");
264     }
265
266     public void updateBalance(String playerName, int amount) {
267         GUI.setBalance(playerName, amount);
268     }
269
270     public void insufficiantFunds(String fieldName, String playerName, int balance) {
271         GUI.showMessage(fieldName
272                 + " er ejet af "
273                 + playerName
274                 + ", men du skal betale mere end du har, derfor vil resten af dine penge
    blevet flyttet, Dette er "
275                 + balance
276                 + "kr som "
277                 + playerName + " vil modtage.");
278     }
279
280     public void bonusMessage(String name, int bonus) {
281         GUI.showMessage("du er landet på " + name + " og modtager " + bonus + " Kr.");
282     }
283
284     public void startMessage(String name) {
285         GUI.showMessage("Du er landet på " + name
286                 + ". Slap af indtil næste tur.");
287     }
288
289     public void taxMessageNoOption(int price) {
290             GUI.showMessage("Du har betalt " + price + " i tax");
291             }
292
293     public void insufficiantFundsTax() {
294         GUI.showMessage("Du skylder mere i skat end du har. "
295                 + "Resten af dine værdier vil bliver overført til banken.");
296     }
297     public void showJailForcedPay() {
298         GUI.showMessage("Du har prøvet at komme ud af fængslet 3 gange, uden held - du
    betaler 1000kr for at komme ud"
299                         + " du får ikke lov til at rykke denne tur");
300     }
301     public void showJailTurn() {
302         GUI.showMessage("Slå med terningerne for at prøve at komme ud af fængslet");
```

```
303     }
304
305     public boolean taxPick(String name) {
306         return GUI.getUserLeftButtonPressed("Du er landet på "+ name + " og skal betale
    indkomstskat. "
307                                 + "vil du helst betale 4000 eller 10% af dine totale
    værdier \n (værdi af grunde, huse og kontanter)?","10%", "4000");
308             }
309
310     public void messageTax10percent() {
311         GUI.showMessage("Du har betalt 10% af dine totale værdier");
312             }
313
314     public void taxFunds() {
315         GUI.showMessage("Den skat du skal betale er større end den mængde penge du har og
    du betaler derfor resten af dine penge.");
316     }
317     public String startOfTurn(Player[] player, int currentPlayer){
318         return GUI.getUserButtonPressed("Det er starten af din tur, " +
    player[currentPlayer].getName() + " hvad ønsker du at gøre?", "Køb hus", "Sælg hus", "Rul
    Terninger");
319
320     }
321
322     public String offerToBuy(String possibleBuild){
323         return GUI.getUserButtonPressed("Du ejer nok grunde af en farve til at bygge huse.
    Ønsker du at bygger på en af disse grunde?" + possibleBuild,"Ja","Nej");
324     }
325     public boolean offerMoreHouses(){
326         return GUI.getUserLeftButtonPressed("Ønsker du at købe flere huse?", "Ja", "Nej");
327     }
328     public String offerToSellHouse(String[] sellOptions){
329         return GUI.getUserSelection("Du kan sælge huse på disse grunde!", sellOptions);
330     }
331     public boolean offerToMoreSellHouses(){
332         return GUI.getUserLeftButtonPressed("Ønsker du at sælge flere huse?", "Ja", "Nej");
333     }
334     public String noHouseToSell(){
335         return GUI.getUserButtonPressed("Du ejer ikke nogle huse som kan sælges", "Okay");
336     }
337
338     public String noHouseToBuy() {
339         return GUI.getUserButtonPressed("Du ejer ikke nogle grunde hvor du kan købe huse",
    "Okay");
340
341     }
342     public String offerToSellPlot(){
343         return GUI.getUserButtonPressed("Ønsker du at sælge dine grunde?","Ja","Nej");
344     }
345
346
347     public boolean jailOptions(Player player) {
348         return GUI.getUserLeftButtonPressed(player.getName()+" sidder i fængsel, "
349                 + "vil du betale 1000kr for at komme ud, eller prøve at slå dig ud af
    fængslet?", "Betal 1000kr", "slå for at komme ud");
350     }
351
352     public String buyRoedovervej(){
353         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Rødovervej for
    kr. 1000?", "Ja", "Nej");
354     }
355     public String buyHvidovervej(){
```

```
356         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Hvidovervej for
    kr. 1000?", "Ja", "Nej");
357     }
358     public String buyRoskildevej(){
359         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Roskildevej for
    kr. 1000?", "Ja", "Nej");
360     }
361     public String buyValbyLanggade(){
362         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Valby Langgade
    for kr. 1000?", "Ja", "Nej");
363     }
364     public String buyAllegade(){
365         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Allégade for kr.
    1000?", "Ja", "Nej");
366     }
367     public String buyFredriksbergAlle(){
368         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Fredriksberg
    Allé for kr. 2000?", "Ja", "Nej");
369     }
370     public String buyBulowsvej(){
371         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Bulowsvej for
    kr. 2000?", "Ja", "Nej");
372     }
373     public String buyGlKongevej(){
374         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Gl Kongevej for
    kr. 2000?", "Ja", "Nej");
375     }
376     public String buyBernstorffsvej(){
377         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Bernstorffsvej
    for kr. 2000?", "Ja", "Nej");
378     }
379     public String buyHellerupvej(){
380         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Hellerupvej for
    kr. 2000?", "Ja", "Nej");
381     }
382     public String buyStrandvejen(){
383         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Strandvejen for
    kr. 2000?", "Ja", "Nej");
384     }
385     public String buyTrianglen(){
386         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Trianglen for
    kr. 3000?", "Ja", "Nej");
387     }
388     public String buyOesterbrogade(){
389         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Østerbrogade for
    kr. 3000?", "Ja", "Nej");
390     }
391     public String buyGroenningen(){
392         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Grønningen for
    kr. 3000?", "Ja", "Nej");
393     }
394     public String buyBredgade(){
395         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Bredgade for kr.
    3000?", "Ja", "Nej");
396     }
397     public String buyKgsNytorv(){
398         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Kgs Nytorv for
    kr. 3000?", "Ja", "Nej");
399     }
400     public String buyIstergade(){
401         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Østergade for
    kr. 3000?", "Ja", "Nej");
```

```
402     }
403     public String buyAmagertorv(){
404         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Amagertorv for
    kr. 4000?", "Ja", "Nej");
405     }
406     public String buyVimmelskaftet(){
407         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Vimmelskaftet
    for kr. 4000?", "Ja", "Nej");
408     }
409     public String buyNygade(){
410         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Nygade for kr.
    4000?", "Ja", "Nej");
411     }
412     public String buyFrederiksberggade(){
413         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på
    Frederiksberggade for kr. 4000?", "Ja", "Nej");
414     }
415     public String buyRaadhuspladsen(){
416         return GUI.getUserButtonPressed("Ønsker du at købe et hus/hotel på Rådhuspladsen
    for kr. 4000?", "Ja", "Nej");
417     }
418     public void setHouse(int fieldNumber, int houseCount){
419         GUI.setHouses(fieldNumber, houseCount);
420     }
421     public void setHotel(int fieldNumber, boolean hasHotel){
422         GUI.setHotel(fieldNumber, hasHotel);
423     }
424
425 }
426
```

```java
package control;

import entity.DiceBox;
import entity.Player;
import fields.GameBoard;
import boundary.GUIController;

public class GameController {

    private int currentPlayer = 0;
    private int playerAmmount = 0;
    private boolean onwards = false;
    private Player[] playerlist;
    private boolean won = false;
    private DiceBox box = new DiceBox();
    private GameBoard gameboard = new GameBoard(box);
    private GUIController GUIC = new GUIController();
    private TurnController TurnC;
    private int lostCount = 0;
    private int mode;

    public GameController(int mode) {
        this.mode = mode;
    }

    public void init() {
        setupGame();
        runGame();
        showWinner();
    }

    public void setupGame() {
        System.out.println(gameboard.toString());
        GUIC.createField();
        // Takes a chosen number and creates that amount of players
        while(!onwards){
        if (mode==1) {
            playerAmmount = GUIC.playerAmount();
            if(playerAmmount<7  && playerAmmount>2)
                onwards=true;
            else
                GUIC.playerAmountError();
        }
        // Accepts 2 players in gamemode 2
        else if (mode==2) {
            playerAmmount = GUIC.playerAmountshowcase();
            if (playerAmmount<7  && playerAmmount>1)
                onwards=true;
            else
                GUIC.playerAmountError();
        }}
        playerlist = new Player[playerAmmount];
        GUIC.createPlayers(playerAmmount, playerlist);
        // Creates Controllers dependent on playerlist
        TurnC = new TurnController(GUIC, gameboard, playerlist, mode);
    }

    public void runGame() {
        // The game continues as long as won equals false
        while (!won) {
            if (!playerlist[currentPlayer].getStatus()) {
                // Runs turn for current Player
```

```
63                  TurnC.runTurn(currentPlayer);
64                  GUIC.newPosition(playerlist[currentPlayer]);
65                  // If a player has lost, adds one to lostCount and reset the players owned
  fields
66                  if (playerlist[currentPlayer].getStatus()) {
67                      GUIC.removePlayer(playerlist, currentPlayer);
68                      gameboard.resetOwnedFields(playerlist[currentPlayer]);
69                      lostCount++;
70
71                      // If only one player is left, won is set to true
72                      if (lostCount == playerAmmount - 1) {
73                          won = true;
74                          GUIC.showWin(playerlist, playerAmmount);
75                      }}}
76              // Changes player
77              changePlayer();
78          }
79      }
80      // Method that changes turn
81      public void changePlayer() {
82          if (currentPlayer == playerAmmount - 1) {
83              currentPlayer = 0;
84          } else {
85              currentPlayer++;
86          }
87      }
88      public void showWinner() {
89          GUIC.showWin(playerlist, playerlist.length);
90      }
91
92  }
93
94
```

```java
package control;

import boundary.GUIController;
import entity.DiceBox;
import entity.Player;
import fields.GameBoard;

public class TurnController {
    private GUIController GUIC;
    private GameBoard board;
    private DiceBox box = new DiceBox();
    private Player[] playerlist;
    private FieldController FC;
    private HouseController houseC;
    String choiceofTurn;
    private int mode;

    // for testing only
    private int k = 0;

    public TurnController(GUIController GUIC, GameBoard board, Player[] playerlist, int mode) {
        this.GUIC = GUIC;
        this.board = board;
        this.playerlist = playerlist;
        this.mode = mode;
        FC = new FieldController(GUIC, board, playerlist);
        houseC = new HouseController(GUIC, board, playerlist);
    }

    public void runTurn(int currentPlayer) {
        if (playerlist[currentPlayer].isJailed())
            runJailTurn(currentPlayer);
        else
            runNormalTurn(currentPlayer);
    }

    public void runJailTurn(int currentPlayer){
        // If player has a getoutofjailcard
        if (playerlist[currentPlayer].hasOutofjailcard()) {
            exitCard(currentPlayer);
            runNormalTurn(currentPlayer);
        }
        // If player pays for exit
        else if (GUIC.jailOptions(playerlist[currentPlayer])) {
            exitPay(currentPlayer);
            runNormalTurn(currentPlayer);
        }
        // If player wants to throw the dice for exit
        else
            exitThrow(currentPlayer);
    }

    // Standard turn
    public void runNormalTurn(int currentPlayer) {
        int count = 0;
        boolean run = true;
        optionsStartOfTurn(currentPlayer);
        while(run){
        if (count==2)
            GUIC.twoPair();
        else if (count==1)
```

```
62              GUIC.onePair();
63          // For demo or normal game run
64          if (mode==1){
65              box.rollDice();
66              GUIC.showDice(box.getDice1(), box.getDice2());
67          } else if (mode==2) {
68              box.setDice(0, 1);
69              box.setDice(1, 0);
70              GUIC.showDice(box.getDice1(), 1);
71          }
72
73          if(box.getSum() + playerlist[currentPlayer].getPosition() >= 40) {
74              GUIC.pastStart();
75              GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
76          }
77          if (count !=3) {
78          GUIC.updatePosition(playerlist, currentPlayer, box.getSum());
79          FC.landOnField(currentPlayer);
80          }
81          if (playerlist[currentPlayer].isJailed())
82              run = false;
83          else if (box.isEqual()){
84              count ++;
85          } else
86              run = false;
87          if (count>=3){
88              run = false;
89              playerlist[currentPlayer].setJailed(true);
90              playerlist[currentPlayer].setPosition(10);
91              GUIC.threePair();
92              GUIC.newPosition(playerlist[currentPlayer]);
93          }
94          }
95          }
96
97      // Tur, efter exit fra jail
98      public void afterJailTurn(int currentPlayer) {
99          GUIC.afterJail();
100         GUIC.newPosition(playerlist[currentPlayer]);
101         FC.landOnField(currentPlayer);
102     }
103
104
105     // runtypes in runJailTurn()
106     public void exitCard(int currentPlayer) {
107         playerlist[currentPlayer].setJailed(false);
108         playerlist[currentPlayer].setJailcount(0);
109     }
110
111     public void exitPay(int currentPlayer) {
112         playerlist[currentPlayer].account.addPoints(-1000);
113         playerlist[currentPlayer].setJailed(false);
114         playerlist[currentPlayer].setJailcount(0);
115     }
116     public void exitThrow(int currentPlayer) {
117         if (playerlist[currentPlayer].getJailcount() < 3) {
118             for (int i=1; i<=3; i++) {
119             GUIC.showJailTurn();
120             box.rollDice();
121             GUIC.showDice(box.getDice1(), box.getDice2());
122             if (box.isEqual()){
```

```
123                 playerlist[currentPlayer].movePosition(box.getSum());
124                 board.getField(playerlist[currentPlayer].getPosition()).landOnField(playerl
    ist[currentPlayer]);
125                 playerlist[currentPlayer].setJailed(false);
126                 playerlist[currentPlayer].setJailcount(0);
127                 afterJailTurn(currentPlayer);
128                 break;
129             }
130             }
131             // If the player didn't get out
132             if (playerlist[currentPlayer].isJailed())
133                 playerlist[currentPlayer].addJailcount();
134             }
135             // If the playerlist[currentPlayer] tried exitting 3 times (forced pay)
136             if (playerlist[currentPlayer].getJailcount() == 3) {
137                 GUIC.showJailForcedPay();
138                 playerlist[currentPlayer].account.addPoints(-1000);
139                 playerlist[currentPlayer].setJailed(false);
140                 playerlist[currentPlayer].setJailcount(0);
141     }
142     }
143     // For testing only
144     // No GUI, Controlable dicebox
145     public void exitThrowTest(int currentPlayer, DiceBox testbox) {
146         if (playerlist[currentPlayer].getJailcount() < 3) {
147             if (testbox.isEqual()){
148                 playerlist[currentPlayer].movePosition(testbox.getSum());
149                 playerlist[currentPlayer].setJailed(false);
150                 playerlist[currentPlayer].setJailcount(0);
151             }
152             else
153                 k++;
154             // If the player didn't get out
155             if (k>=3) {
156                 if (playerlist[currentPlayer].isJailed()) {
157                     playerlist[currentPlayer].addJailcount();
158                     k=0;
159                 }
160             // If the player tried exitting 3 times (forced pay)
161                     if (playerlist[currentPlayer].getJailcount() == 3) {
162                         playerlist[currentPlayer].account.addPoints(-1000);
163                         playerlist[currentPlayer].setJailed(false);
164                         playerlist[currentPlayer].setJailcount(0);
165             }
166             }
167     }
168     }
169     // For testing only
170     public void runNormaltestTurn(int currentPlayer,
171             DiceBox box1, DiceBox box2, DiceBox box3) {
172         int count = 0;
173         boolean run = true;
174         while(run){
175         if(count==0)
176         GUIC.nextPlayer(playerlist, currentPlayer);
177         else if (count==2)
178         GUIC.twoPair();
179         else
180         GUIC.onePair();;
181
182         if (count==0) {
183         GUIC.showDice(box1.getDice1(), box1.getDice2());
```

```
184         GUIC.updatePosition(playerlist, currentPlayer, box1.getSum());
185         } else if (count==1) {
186         GUIC.showDice(box2.getDice1(), box2.getDice2());
187         GUIC.updatePosition(playerlist, currentPlayer, box2.getSum());
188         } else {
189         GUIC.showDice(box3.getDice1(), box3.getDice2());
190         GUIC.updatePosition(playerlist, currentPlayer, box3.getSum());
191         }
192
193         FC.landOnField(currentPlayer);
194         if (box.isEqual()){
195             count ++;
196         } else
197             run = false;
198         if (count>=3){
199             run = false;
200             playerlist[currentPlayer].setJailed(true);
201             playerlist[currentPlayer].setPosition(10);
202             GUIC.threePair();;
203             GUIC.newPosition(playerlist[currentPlayer]);
204         }
205         }
206         }
207
208     public void optionsStartOfTurn(int currentPlayer){
209         boolean keepBuyingSelling = true;
210         while(keepBuyingSelling){
211         choiceofTurn = GUIC.startOfTurn(playerlist, currentPlayer);
212         if(choiceofTurn.equals("Køb hus")){
213             buyHouse(currentPlayer);
214
215         }else if(choiceofTurn.equals("Sælg hus")){
216             sellHouse(currentPlayer);
217
218         }else if(choiceofTurn.equals("Rul Terninger")){
219             keepBuyingSelling = false;
220         }
221         }
222     }
223     public void buyHouse(int currentPlayer){
224         houseC.checkOwnedFields(currentPlayer);
225         if(playerlist[currentPlayer].getBuy_Blue()==
    true||playerlist[currentPlayer].getBuy_Pink()==
    true||playerlist[currentPlayer].getBuy_Green()== true
226                 ||playerlist[currentPlayer].getBuy_grey()==
    true||playerlist[currentPlayer].getBuy_Red()==
    true||playerlist[currentPlayer].getBuy_White() == true
227                 ||playerlist[currentPlayer].getBuy_Yellow()==
    true||playerlist[currentPlayer].getBuy_Magenta()== true){
228                 houseC.buyHouse(currentPlayer);
229         }else{
230             GUIC.noHouseToBuy();
231         }
232     }
233     public void sellHouse(int currentPlayer){
234         String sellHouse;
235         boolean sellMore = true;
236         String[] ar = houseC.checkIfPossibleSell(currentPlayer, board);
237
238         if(playerlist[currentPlayer].getBuy_Blue()==
    true||playerlist[currentPlayer].getBuy_Pink()==
    true||playerlist[currentPlayer].getBuy_Green()== true
```

```
239                 ||playerlist[currentPlayer].getBuy_grey()==
    true||playerlist[currentPlayer].getBuy_Red()==
    true||playerlist[currentPlayer].getBuy_White() == true
240                 ||playerlist[currentPlayer].getBuy_Yellow()==
    true||playerlist[currentPlayer].getBuy_Magenta()== true){
241                 if(ar.length > 0){
242                         while(sellMore == true){
243                             ar = houseC.checkIfPossibleSell(currentPlayer, board);
244                             if(ar.length == 0){
245                                 GUIC.noHouseToSell();
246                                 sellMore = false;
247                                 break;
248                             }
249                             sellHouse =
    GUIC.offerToSellHouse(houseC.checkIfPossibleSell(currentPlayer, board));
250                             houseC.sellHouse(currentPlayer, board, sellHouse);
251                             sellMore = GUIC.offerToMoreSellHouses();
252                         }
253                     }
254                 }else{
255                     GUIC.noHouseToSell();
256                 }
257     }
258 }
259
```

```java
package control;

import boundary.GUIController;
import entity.Player;
import fields.Field;
import fields.GameBoard;


public class HouseController {
    private GUIController GUIC;
    private GameBoard board;
    private Player[] playerlist;
    private boolean khan = true;
    private String[] sellOptions;

    public HouseController(GUIController GUIC, GameBoard board,Player[] playerlist){
        this.board =board;
        this.GUIC = GUIC;
        this.playerlist = playerlist;
        this.board = board;
    }

    public void buyHouse(int currentPlayer){
        //check if you can buy.
        boolean moreHouses = true;
        while(moreHouses== true){
            for(int i=1; i<=8; i++){
                if(getPriceAndValue(currentPlayer, i, board) == true){
                    for(int q=1; q<=8; q++){
                        if(getBuild(q, currentPlayer) == true){
                            buildPlots(currentPlayer ,q);
                            }
                        }
                }
            }

            if(GUIC.offerMoreHouses()==false){
                moreHouses=false;
            }
        }
    }


    //sell house
    public void sellHouse(int currentPlayer,GameBoard board, String plot){
            for(int i=1; i<=39; i++){
                if(board.getField(i).getName().equals(plot)){
                    playerlist[currentPlayer].account.addPoints(getHousePrice(1)/2);
                    board.getField(i).addNumberofHouses(-1);
                    if(board.getField(i).getNumberofhouses()==5){
                        GUIC.setHotel(i+1,false);
                        removeHotel(currentPlayer);
                    }else{
                    playerlist[currentPlayer].addHouseammount(-1);
                    GUIC.setHouse(i+1, board.getField(i).getNumberofhouses());
                    }
                }
            }GUIC.updateBalance(playerlist[currentPlayer].getName(),
  playerlist[currentPlayer].account.getScore());
        }
    public String[] checkIfPossibleSell(int currentPlayer, GameBoard board){
        int arrayIndex = 0;
```

```
62          int arraylength = 0;
63          for(int i=1; i<=39; i++){
64              if(board.getField(i).getNumberofhouses() > 0 &&
  playerlist[currentPlayer].equals(board.getField(i).getOwner())){
65                  arraylength++;
66              }
67          }
68          sellOptions = new String[arraylength];
69          for(int i=1; i<=39; i++){
70              if(board.getField(i).getNumberofhouses() > 0 &&
  playerlist[currentPlayer].equals(board.getField(i).getOwner())){
71                  sellOptions[arrayIndex] = board.getField(i).getName();
72                  arrayIndex++;
73              }
74          }
75              return sellOptions;
76      }
77
78      public void checkOwnedFields(int currentPlayer){
79          checkBlue(currentPlayer);
80          checkPink(currentPlayer);
81          checkGreen(currentPlayer);
82          checkGrey(currentPlayer);
83          checkRed(currentPlayer);
84          checkWhite(currentPlayer);
85          checkYellow(currentPlayer);
86          checkMagneta(currentPlayer);
87      }
88      public void checkBlue(int currentPlayer){
89          if(playerlist[currentPlayer].getFieldammount_blue() == 2){
90              playerlist[currentPlayer].setBuy_Blue(khan);
91          }
92      }
93
94      public void checkPink(int currentPlayer){
95          if(playerlist[currentPlayer].getFieldammount_pink() == 3){
96              playerlist[currentPlayer].setBuy_Pink(khan);
97          }
98      }
99
100     public void checkGreen(int currentPlayer){
101         if(playerlist[currentPlayer].getFieldammount_green() == 3){
102             playerlist[currentPlayer].setBuy_Green(khan);
103         }
104     }
105
106     public void checkGrey(int currentPlayer){
107         if(playerlist[currentPlayer].getFieldammount_grey() == 3){
108             playerlist[currentPlayer].setBuy_grey(khan);
109         }
110     }
111
112     public void checkRed(int currentPlayer){
113         if(playerlist[currentPlayer].getFieldammount_red() == 3){
114             playerlist[currentPlayer].setBuy_Red(khan);
115         }
116     }
117
118     public void checkWhite(int currentPlayer){
119         if(playerlist[currentPlayer].getFieldammount_white() == 3){
120             playerlist[currentPlayer].setBuy_White(khan);
121         }
```

```
122     }
123
124     public void checkYellow(int currentPlayer){
125         if(playerlist[currentPlayer].getFieldammount_yellow() == 3){
126             playerlist[currentPlayer].setBuy_Yellow(khan);
127         }
128     }
129
130     public void checkMagneta(int currentPlayer){
131         if(playerlist[currentPlayer].getFieldammount_magenta() == 2){
132             playerlist[currentPlayer].setBuy_Magenta(khan);
133         }
134     }
135
136     public boolean getBuild(int n, int currentPlayer){
137         if(n == 1){
138             return playerlist[currentPlayer].getBuy_Blue();
139         }
140         else if(n == 2){
141             return playerlist[currentPlayer].getBuy_Pink();
142         }
143         else if(n == 3){
144             return playerlist[currentPlayer].getBuy_Green();
145         }
146         else if(n == 4){
147             return playerlist[currentPlayer].getBuy_grey();
148         }
149         else if(n == 5){
150             return playerlist[currentPlayer].getBuy_Red();
151         }
152         else if(n == 6){
153             return playerlist[currentPlayer].getBuy_White();
154         }
155         else if(n == 7){
156             return playerlist[currentPlayer].getBuy_Yellow();
157         }
158         else if(n == 8){
159             return playerlist[currentPlayer].getBuy_Magenta();
160         }
161         return false;
162     }
163
164     public boolean getPriceAndValue(int currentPlayer, int n, GameBoard board){
165         if(n == 1){
166             if(playerlist[currentPlayer].getBuy_Blue() == true){
167                 if(playerlist[currentPlayer].account.getScore() >=
    board.getField(playerlist[currentPlayer].getPosition()).getPrice()){
168                     return true;
169                 }
170             }
171         }
172         else if(n == 2){
173             if(playerlist[currentPlayer].getBuy_Pink() == true){
174                 if(playerlist[currentPlayer].account.getScore() >=
    board.getField(playerlist[currentPlayer].getPosition()).getPrice()){
175                     return true;
176                 }
177             }
178         }
179         else if(n == 3){
180             if(playerlist[currentPlayer].getBuy_Green() == true){
181                 if(playerlist[currentPlayer].account.getScore() >=
```

```
   board.getField(playerlist[currentPlayer].getPosition()).getPrice()){
182                     return true;
183                 }
184             }
185         }
186         else if(n == 4){
187             if(playerlist[currentPlayer].getBuy_grey() == true){
188                 if(playerlist[currentPlayer].account.getScore() >=
   board.getField(playerlist[currentPlayer].getPosition()).getPrice()){
189                     return true;
190                 }
191             }
192         }
193         else if(n == 5){
194             if(playerlist[currentPlayer].getBuy_Red() == true){
195                 if(playerlist[currentPlayer].account.getScore() >=
   board.getField(playerlist[currentPlayer].getPosition()).getPrice()){
196                     return true;
197                 }
198             }
199         }
200         else if(n == 6){
201             if(playerlist[currentPlayer].getBuy_White() == true){
202                 if(playerlist[currentPlayer].account.getScore() >=
   board.getField(playerlist[currentPlayer].getPosition()).getPrice()){
203                     return true;
204                 }
205             }
206         }
207         else if(n == 7){
208             if(playerlist[currentPlayer].getBuy_Yellow() == true){
209                 if(playerlist[currentPlayer].account.getScore() >=
   board.getField(playerlist[currentPlayer].getPosition()).getPrice()){
210                     return true;
211                 }
212             }
213         }
214         else if(n == 8){
215             if(playerlist[currentPlayer].getBuy_Magenta() == true){
216                 if(playerlist[currentPlayer].account.getScore() >=
   board.getField(playerlist[currentPlayer].getPosition()).getPrice()){
217                     return true;
218                 }
219             }
220         }
221 
222         return false;
223 
224     }
225 
226     public void setHotel(int currentPlayer){
227         playerlist[currentPlayer].addHouseammount(-4);
228         playerlist[currentPlayer].addHotelammount(1);
229     }
230 
231     public void removeHotel(int currentPlayer){
232         playerlist[currentPlayer].addHouseammount(4);
233         playerlist[currentPlayer].addHotelammount(-1);
234     }
235 
236     public int getHousePrice(int n){
237         return board.getField(n).getHouseprice();
```

```
238     }
239 
240     public void buildPlots(int currentPlayer, int n){
241         if(n == 1){
242             Field f1 = board.getField(1);
243             Field f2 = board.getField(3);
244 
245             if(f1.getNumberofhouses() <= f2.getNumberofhouses()){
246                 if(f1.getNumberofhouses()<5){
247                     if(GUIC.buyRoedovervej().equals("Ja")){
248                         playerlist[currentPlayer].account.addPoints(-getHousePrice(1));
249                         f1.addNumberofHouses(1);
250                         if(f1.getNumberofhouses()==5){
251                             GUIC.setHotel(2,true);
252                             setHotel(currentPlayer);
253                         }else{
254                         GUIC.setHouse(2, f1.getNumberofhouses());
255                         playerlist[currentPlayer].addHouseammount(1);
256                         }
257                     }}
258                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
259                 }
260             if(f2.getNumberofhouses() <= f1.getNumberofhouses()){
261                 if(f2.getNumberofhouses()<5){
262                     if(GUIC.buyHvidovervej().equals("Ja")){
263                         playerlist[currentPlayer].account.addPoints(-getHousePrice(3));
264                         f2.addNumberofHouses(1);
265                         if(f2.getNumberofhouses()==5){
266                             GUIC.setHotel(4,true);
267                             setHotel(currentPlayer);
268                         }else{
269                         GUIC.setHouse(4, f2.getNumberofhouses());
270                         playerlist[currentPlayer].addHouseammount(1);
271                         }
272                     }}}
273             GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
274         }
275         if(n == 2){
276             Field f3 = board.getField(6);
277             Field f4 = board.getField(8);
278             Field f5 = board.getField(9);
279             if(f3.getNumberofhouses() <= f4.getNumberofhouses() && f3.getNumberofhouses()
    <= f5.getNumberofhouses()){
280                 if(f3.getNumberofhouses()<5){
281                     if(GUIC.buyRoskildevej().equals("Ja")){
282                         playerlist[currentPlayer].account.addPoints(-getHousePrice(6));
283                         f3.addNumberofHouses(1);
284                         if(f3.getNumberofhouses()==5){
285                             GUIC.setHotel(7,true);
286                             setHotel(currentPlayer);
287                         }else{
288                         GUIC.setHouse(7, f3.getNumberofhouses());
289                         playerlist[currentPlayer].addHouseammount(1);
290                         }}}
291                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
292                 }
293             if(f4.getNumberofhouses() <= f3.getNumberofhouses() && f4.getNumberofhouses()
    <= f5.getNumberofhouses()){
294                 if(f4.getNumberofhouses()<5){
```

```
295                     if(GUIC.buyValbyLanggade().equals("Ja")){
296                         playerlist[currentPlayer].account.addPoints(-getHousePrice(8));
297                         f4.addNumberofHouses(1);
298                         if(f4.getNumberofhouses()==5){
299                             GUIC.setHotel(9,true);
300                             setHotel(currentPlayer);
301                         }else{
302                         GUIC.setHouse(9, f4.getNumberofhouses());
303                         playerlist[currentPlayer].addHouseammount(1);
304                         }}}}
305             if(f5.getNumberofhouses() <= f3.getNumberofhouses() && f5.getNumberofhouses()
    <= f4.getNumberofhouses()){
306                 if(f5.getNumberofhouses()<5){
307                     if(GUIC.buyAllegade().equals("Ja")){
308                         playerlist[currentPlayer].account.addPoints(-getHousePrice(9));
309                         f5.addNumberofHouses(1);
310                         if(f5.getNumberofhouses()==5){
311                             GUIC.setHotel(10,true);
312                             setHotel(currentPlayer);
313                         }else{
314                         GUIC.setHouse(10, f5.getNumberofhouses());
315                         playerlist[currentPlayer].addHouseammount(1);
316                         }}}
317                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
318                 }
319         }
320         if(n == 3){
321             Field f6 = board.getField(11);
322             Field f7 = board.getField(13);
323             Field f8 = board.getField(14);
324 
325             if(f6.getNumberofhouses() <= f7.getNumberofhouses() && f6.getNumberofhouses()
    <= f8.getNumberofhouses()){
326                 if(f6.getNumberofhouses()<5){
327                     if(GUIC.buyFredriksbergAlle().equals("Ja")){
328                         playerlist[currentPlayer].account.addPoints(-getHousePrice(11));
329                         f6.addNumberofHouses(1);
330                         if(f6.getNumberofhouses()==5){
331                             GUIC.setHotel(12,true);
332                             setHotel(currentPlayer);
333                         }else{
334                         GUIC.setHouse(12, f6.getNumberofhouses());
335                         playerlist[currentPlayer].addHouseammount(1);
336                         }}}
337                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
338                 }
339             if(f7.getNumberofhouses() <= f6.getNumberofhouses() && f7.getNumberofhouses()
    <= f8.getNumberofhouses()){
340                 if(f7.getNumberofhouses()<5){
341                     if(GUIC.buyBulowsvej().equals("Ja")){
342                         playerlist[currentPlayer].account.addPoints(-getHousePrice(13));
343                         f7.addNumberofHouses(1);
344                         if(f7.getNumberofhouses()==5){
345                             GUIC.setHotel(14,true);
346                             setHotel(currentPlayer);
347                         }else{
348                         GUIC.setHouse(14, f7.getNumberofhouses());
349                         playerlist[currentPlayer].addHouseammount(1);
350                         }}}
351                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
```

```
    playerlist[currentPlayer].account.getScore());
352                 }
353             if(f8.getNumberofhouses() <= f6.getNumberofhouses() && f8.getNumberofhouses()
    <= f7.getNumberofhouses()){
354                 if(f8.getNumberofhouses()<5){
355                     if(GUIC.buyGlKongevej().equals("Ja")){
356                         playerlist[currentPlayer].account.addPoints(-getHousePrice(14));
357                         f8.addNumberofHouses(1);
358                         if(f8.getNumberofhouses()==5){
359                             GUIC.setHotel(15,true);
360                             setHotel(currentPlayer);
361                         }else{
362                         GUIC.setHouse(15, f8.getNumberofhouses());
363                         playerlist[currentPlayer].addHouseammount(1);
364                         }}}
365                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
366                 }
367         }
368         if(n == 4){
369             Field f9 = board.getField(16);
370             Field f10 = board.getField(18);
371             Field f11 = board.getField(19);
372 
373             if(f9.getNumberofhouses() <= f10.getNumberofhouses() && f9.getNumberofhouses()
    <= f11.getNumberofhouses()){
374                 if(f9.getNumberofhouses()<5){
375                     if(GUIC.buyBernstorffsvej().equals("Ja")){
376                         playerlist[currentPlayer].account.addPoints(-getHousePrice(16));
377                         f9.addNumberofHouses(1);
378                         if(f9.getNumberofhouses()==5){
379                             GUIC.setHotel(17,true);
380                             setHotel(currentPlayer);
381                         }else{
382                         GUIC.setHouse(17, f9.getNumberofhouses());
383                         playerlist[currentPlayer].addHouseammount(1);
384                         }}}
385                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
386                 }
387             if(f10.getNumberofhouses() <= f9.getNumberofhouses() && f10.getNumberofhouses()
    <= f11.getNumberofhouses()){
388                 if(f10.getNumberofhouses()<5){
389                     if(GUIC.buyHellerupvej().equals("Ja")){
390                         playerlist[currentPlayer].account.addPoints(-getHousePrice(18));
391                         f10.addNumberofHouses(1);
392                         if(f10.getNumberofhouses()==5){
393                             GUIC.setHotel(19,true);
394                             setHotel(currentPlayer);
395                         }else{
396                         GUIC.setHouse(19, f10.getNumberofhouses());
397                         playerlist[currentPlayer].addHouseammount(1);
398                         }}}
399                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
400                 }
401             if(f11.getNumberofhouses() <= f9.getNumberofhouses() && f11.getNumberofhouses()
    <= f10.getNumberofhouses()){
402                 if(f11.getNumberofhouses()<5){
403                     if(GUIC.buyStrandvejen().equals("Ja")){
404                         playerlist[currentPlayer].account.addPoints(-getHousePrice(19));
405                         f11.addNumberofHouses(1);
```

```
406                         if(f11.getNumberofhouses()==5){
407                             GUIC.setHotel(20,true);
408                             setHotel(currentPlayer);
409                         }else{
410                         GUIC.setHouse(20, f11.getNumberofhouses());
411                         playerlist[currentPlayer].addHouseammount(1);
412                         }}}
413                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
414                         }
415         }
416 
417         if(n == 5){
418             Field f12 = board.getField(21);
419             Field f13 = board.getField(23);
420             Field f14 = board.getField(24);
421 
422             if(f12.getHouseprice() <= f13.getNumberofhouses() && f12.getHouseprice() <=
    f14.getNumberofhouses()){
423                 if(f12.getNumberofhouses()<5){
424                     if(GUIC.buyTrianglen().equals("Ja")){
425                         playerlist[currentPlayer].account.addPoints(-getHousePrice(21));
426                         f12.addNumberofHouses(1);
427                         if(f12.getNumberofhouses()==5){
428                             GUIC.setHotel(22,true);
429                             setHotel(currentPlayer);
430                         }else{
431                         GUIC.setHouse(22, f12.getNumberofhouses());
432                         playerlist[currentPlayer].addHouseammount(1);
433                         }}}
434                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
435                 }
436             if(f13.getNumberofhouses() <= f12.getNumberofhouses() &&
    f13.getNumberofhouses() <= f14.getNumberofhouses()){
437                 if(f13.getNumberofhouses()<5){
438                     if(GUIC.buyOesterbrogade().equals("Ja")){
439                         playerlist[currentPlayer].account.addPoints(-getHousePrice(23));
440                         f13.addNumberofHouses(1);
441                         if(f13.getNumberofhouses()==5){
442                             GUIC.setHotel(24,true);
443                             setHotel(currentPlayer);
444                         }else{
445                         GUIC.setHouse(24, f13.getNumberofhouses());
446                         playerlist[currentPlayer].addHouseammount(1);
447                         }}}
448                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
449                         }
450             if(f14.getNumberofhouses() <= f12.getNumberofhouses() &&
    f14.getNumberofhouses() <= f13.getNumberofhouses()){
451                 if(f14.getNumberofhouses()<5){
452                     if(GUIC.buyGroenningen().equals("Ja")){
453                         playerlist[currentPlayer].account.addPoints(-getHousePrice(24));
454                         f14.addNumberofHouses(1);
455                         if(f14.getNumberofhouses()==5){
456                             GUIC.setHotel(25,true);
457                             setHotel(currentPlayer);
458                         }else{
459                         GUIC.setHouse(25, f14.getNumberofhouses());
460                         playerlist[currentPlayer].addHouseammount(1);
461                         }}}
```

```
462                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
463                 }
464         }
465 
466         if(n == 6){
467             Field f15 = board.getField(26);
468             Field f16 = board.getField(27);
469             Field f17 = board.getField(29);
470             if(f15.getNumberofhouses() <= f16.getNumberofhouses() &&
    f15.getNumberofhouses() <= f17.getNumberofhouses()){
471                 if(f15.getNumberofhouses()<5){
472                     if(GUIC.buyBredgade().equals("Ja")){
473                         playerlist[currentPlayer].account.addPoints(-getHousePrice(26));
474                         f15.addNumberofHouses(1);
475                         if(f15.getNumberofhouses()==5){
476                             GUIC.setHotel(27,true);
477                             setHotel(currentPlayer);
478                         }else{
479                         GUIC.setHouse(27, f15.getNumberofhouses());
480                         playerlist[currentPlayer].addHouseammount(1);
481                         }}}
482                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
483                 }
484             if(f16.getNumberofhouses() <= f15.getNumberofhouses() &&
    f16.getNumberofhouses() <= f17.getNumberofhouses()){
485                 if(f16.getNumberofhouses()<5){
486                     if(GUIC.buyKgsNytorv().equals("Ja")){
487                         playerlist[currentPlayer].account.addPoints(-getHousePrice(27));
488                         f16.addNumberofHouses(1);
489                         if(f16.getNumberofhouses()==5){
490                             GUIC.setHotel(28,true);
491                             setHotel(currentPlayer);
492                         }else{
493                         GUIC.setHouse(28, f16.getNumberofhouses());
494                         playerlist[currentPlayer].addHouseammount(1);
495                         }}}
496                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
497                 }
498             if(f17.getNumberofhouses() <= f15.getNumberofhouses() &&
    f17.getNumberofhouses() <= f16.getNumberofhouses()){
499                 if(f17.getNumberofhouses()<5){
500                     if(GUIC.buyIstergade().equals("Ja")){
501                         playerlist[currentPlayer].account.addPoints(-getHousePrice(29));
502                         f17.addNumberofHouses(1);
503                         if(f17.getNumberofhouses()==5){
504                             GUIC.setHotel(30,true);
505                             setHotel(currentPlayer);
506                         }else{
507                         GUIC.setHouse(30, f17.getNumberofhouses());
508                         playerlist[currentPlayer].addHouseammount(1);
509                         }}}
510                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
511                 }
512         }
513 
514         if(n == 7){
515             Field f18 = board.getField(31);
516             Field f19 = board.getField(32);
```

```
517             Field f20 = board.getField(34);
518 
519             if(f18.getNumberofhouses() <= f19.getNumberofhouses() &&
    f18.getNumberofhouses() <= f20.getNumberofhouses()){
520                 if(f18.getNumberofhouses()<5){
521                     if(GUIC.buyAmagertorv().equals("Ja")){
522                         playerlist[currentPlayer].account.addPoints(-getHousePrice(31));
523                         f18.addNumberofHouses(1);
524                         if(f18.getNumberofhouses()==5){
525                             GUIC.setHotel(32,true);
526                             setHotel(currentPlayer);
527                         }else{
528                             GUIC.setHouse(32, f18.getNumberofhouses());
529                             playerlist[currentPlayer].addHouseammount(1);
530                             }}}
531                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
532                 }
533             if(f19.getNumberofhouses() <= f18.getNumberofhouses() &&
    f19.getNumberofhouses() <= f20.getNumberofhouses()){
534                 if(f19.getNumberofhouses()<5){
535                     if(GUIC.buyVimmelskaftet().equals("Ja")){
536                         playerlist[currentPlayer].account.addPoints(-getHousePrice(32));
537                         f19.addNumberofHouses(1);
538                         if(f19.getNumberofhouses()==5){
539                             GUIC.setHotel(33,true);
540                             setHotel(currentPlayer);
541                         }else{
542                         GUIC.setHouse(33, f19.getNumberofhouses());
543                         playerlist[currentPlayer].addHouseammount(1);
544                         }}}
545                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
546                 }
547             if(f20.getNumberofhouses() <= f18.getNumberofhouses() &&
    f20.getNumberofhouses() <= f19.getNumberofhouses()){
548                 if(f20.getNumberofhouses()<5){
549                     if(GUIC.buyNygade().equals("Ja")){
550                         playerlist[currentPlayer].account.addPoints(-getHousePrice(34));
551                         f20.addNumberofHouses(1);
552                         if(f20.getNumberofhouses()==5){
553                             GUIC.setHotel(35,true);
554                             setHotel(currentPlayer);
555                         }else{
556                         GUIC.setHouse(35, f20.getNumberofhouses());
557                         playerlist[currentPlayer].addHouseammount(1);
558                         }}}
559                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
560                 }
561         }
562         if(n == 8){
563             Field f21 = board.getField(37);
564             Field f22 = board.getField(39);
565 
566             if(f21.getNumberofhouses() <= f22.getNumberofhouses()){
567                 if(f21.getNumberofhouses()<5){
568                     if(GUIC.buyFrederiksberggade().equals("Ja")){
569                         playerlist[currentPlayer].account.addPoints(-getHousePrice(37));
570                         f21.addNumberofHouses(1);
571                         if(f21.getNumberofhouses()==5){
572                             GUIC.setHotel(38,true);
```

```
573                             setHotel(currentPlayer);
574                         }else{
575                         GUIC.setHouse(38, f21.getNumberofhouses());
576                         playerlist[currentPlayer].addHouseammount(1);
577                         }}}
578                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
579                 }
580             if(f22.getNumberofhouses() <= f21.getNumberofhouses()){
581                 if(f22.getNumberofhouses()<5){
582                     if(GUIC.buyRaadhuspladsen().equals("Ja")){
583                         playerlist[currentPlayer].account.addPoints(-getHousePrice(39));
584                         f22.addNumberofHouses(1);
585                         if(f22.getNumberofhouses()==5){
586                             GUIC.setHotel(40,true);
587                             setHotel(currentPlayer);
588                         }else{
589                         GUIC.setHouse(40, f22.getNumberofhouses());
590                         playerlist[currentPlayer].addHouseammount(1);
591                         }}}
592                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
593                 }
594         }
595     }
596
597 }
598
599
600
```

```java
package control;


import entity.Player;
import fields.GameBoard;
import boundary.GUIController;

public class FieldController {
    private GUIController GUIC;
    private GameBoard gameboard;
    private Player[] playerlist;
    private DeckController DC;

    public FieldController(GUIController GUIC, GameBoard gameboard, Player[] playerlist) {
        this.GUIC = GUIC;
        this.gameboard = gameboard;
        this.playerlist = playerlist;
        DC = new DeckController(GUIC, playerlist, gameboard, this);
    }

    public void landOnField(int currentPlayer) {
        // For Territories
        if (gameboard.getField(playerlist[currentPlayer].getPosition()) instanceof fields.Territory)
            landOnTerritory(currentPlayer);
        // For Fleets
        else if (gameboard.getField(playerlist[currentPlayer].getPosition()) instanceof fields.Fleet)
            landOnFleet(currentPlayer);
        // For LaborCamps
        else if (gameboard.getField(playerlist[currentPlayer].getPosition()) instanceof fields.LaborCamp)
            landOnLaborCamp(currentPlayer);
        // For ChanceCard
        else if (gameboard.getField(playerlist[currentPlayer].getPosition()) instanceof fields.ChanceCard)
            DC.drawCard(currentPlayer);
        // For MoveToJail
        else if (gameboard.getField(playerlist[currentPlayer].getPosition()) instanceof fields.MoveToJail) {
            GUIC.moveToJail();
            gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(playerlist[currentPlayer]);
            GUIC.newPosition(playerlist[currentPlayer]);
        }
        // For Tax
        else if (gameboard.getField(playerlist[currentPlayer].getPosition()) instanceof fields.Tax)
            landOnTax(currentPlayer);
        // For Refuge
        else if (gameboard.getField(playerlist[currentPlayer].getPosition()) instanceof fields.Refuge)
            landOnRefuge(currentPlayer);
        // For Start
        else if (gameboard.getField(playerlist[currentPlayer].getPosition()) instanceof fields.Start)
            GUIC.startMessage(gameboard.getField(playerlist[currentPlayer].getPosition()).getName());
        // For every other fields
        else
            gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(playerlist[currentPlayer]);
```

```
52
53          // Update on GUI
54          for (int i = 0; i<playerlist.length; i++) {
55          GUIC.newPosition(playerlist[i]);
56          GUIC.updateBalance(playerlist[i].getName(), playerlist[i].account.getScore());
57          }
58      }
59
60
61      public void landOnTerritory(int currentPlayer) {
62          // If no owner
63          if (gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner() == null)
  {
64              // If affordable
65              if (playerlist[currentPlayer].account.getScore() >=
  gameboard.getField(playerlist[currentPlayer].getPosition()).getPrice()) {
66                  // Player chooses if he wants to buy the field
67                  boolean buyfield =
  GUIC.buyField(gameboard.getField(playerlist[currentPlayer].getPosition()).getName(),
68                  gameboard.getField(playerlist[currentPlayer].getPosition()).getPrice());
69                  gameboard.getField(playerlist[currentPlayer].getPosition()).setBuyfield(buy
  field);
70                  gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
  yerlist[currentPlayer]);
71
72                  // Field status on GUI
73                  if (buyfield) {
74                      GUIC.fieldBought(gameboard.getField(playerlist[currentPlayer].getPositi
  on()).getName());
75                      GUIC.setOwner(playerlist[currentPlayer]);
76                      GUIC.updateBalance(playerlist[currentPlayer].getName(),
  playerlist[currentPlayer].account.getScore());
77                  } else
78                      GUIC.fieldRefused(gameboard.getField(playerlist[currentPlayer].getPosit
  ion()).getName());
79                  } else
80                      GUIC.fieldRefusedPrice(gameboard.getField(playerlist[currentPlayer].get
  Position()).getName());
81          }
82          // If its your own field
83          else if (gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner() ==
  playerlist[currentPlayer]) {
84              GUIC.fieldOwnedByPlayer(gameboard.getField(playerlist[currentPlayer].getPositio
  n()).getName());
85          }
86          // If someone else owns the field
87          else {
88              // If affordable rent
89              if
  (gameboard.getField(playerlist[currentPlayer].getPosition()).checkPayDoubleRent(playerlist[c
  urrentPlayer])) {
90                  if(playerlist[currentPlayer].account.getScore() >=
  gameboard.getField(playerlist[currentPlayer].getPosition()).getRent(0)*2)
91                      GUIC.fieldTax(gameboard.getField(playerlist[currentPlayer].getPosition(
  )).getName(),
92                              gameboard.getField(playerlist[currentPlayer].getPosition()).get
  Owner().getName(),
93                              gameboard.getField(playerlist[currentPlayer].getPosition()).get
  Rent(0)*2);
94                  else
95                      GUIC.insufficiantFunds(gameboard.getField(playerlist[currentPlayer].get
  Position()).getName(),
```

```
 96                             gameboard.getField(playerlist[currentPlayer].getPosition()).get
    Owner().getName(),
 97                             playerlist[currentPlayer].account.getScore());
 98                     gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(
    playerlist[currentPlayer]);
 99
100             }else if (playerlist[currentPlayer].account.getScore() >=
101                     gameboard.getField(playerlist[currentPlayer].getPosition()).getRent(
102                             gameboard.getField(playerlist[currentPlayer].getPosition()).get
    Numberofhouses())){
103                 GUIC.fieldTax(gameboard.getField(playerlist[currentPlayer].getPosition()).g
    etName(),
104                         gameboard.getField(playerlist[currentPlayer].getPosition()).getOwne
    r().getName(),
105                         gameboard.getField(playerlist[currentPlayer].getPosition()).getRent
106                         (gameboard.getField(playerlist[currentPlayer].getPosition()).getNum
    berofhouses()));;
107                         gameboard.getField(playerlist[currentPlayer].getPosition()).landOnF
    ield(playerlist[currentPlayer]);
108             // if insufficient funds
109             } else {
110                 GUIC.insufficiantFunds(gameboard.getField(playerlist[currentPlayer].getPosi
    tion()).getName(),
111                         gameboard.getField(playerlist[currentPlayer].getPosition()).getOwne
    r().getName(),
112                         playerlist[currentPlayer].account.getScore());
113                 gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
    yerlist[currentPlayer]);
114             }
115
116             GUIC.updateBalance(gameboard.getField(playerlist[currentPlayer].getPosition()).g
    etOwner().getName(),
117                     gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner().a
    ccount.getScore());
118             GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
119
120         }
121     }
122
123     public void landOnFleet(int currentPlayer) {
124         // If no owner
125         if (gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner() == null)
    {
126             // If affordable
127             if (playerlist[currentPlayer].account.getScore() >=
    gameboard.getField(playerlist[currentPlayer].getPosition()).getPrice()) {
128                 // Player chooses if he wants to buy the field
129                 boolean buyfield =
    GUIC.buyField(gameboard.getField(playerlist[currentPlayer].getPosition()).getName(),
130                         gameboard.getField(playerlist[currentPlayer].getPosition()).getPric
    e());
131                 gameboard.getField(playerlist[currentPlayer].getPosition()).setBuyfield(buy
    field);
132                 gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
    yerlist[currentPlayer]);
133
134                 // Field status on GUI
135                 if (buyfield) {
136                     GUIC.fieldBought(gameboard.getField(playerlist[currentPlayer].getPositi
    on()).getName());
137                     GUIC.setOwner(playerlist[currentPlayer]);
```

```
138                     GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
139                 } else
140                     GUIC.fieldRefused(gameboard.getField(playerlist[currentPlayer].getPosit
    ion()).getName());
141                 } else
142                     GUIC.fieldRefusedPrice(gameboard.getField(playerlist[currentPlayer].get
    Position()).getName());
143         }
144         // If its your own field
145         else if (gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner() ==
    playerlist[currentPlayer]) {
146             GUIC.fieldOwnedByPlayer(gameboard.getField(playerlist[currentPlayer].getPositio
    n()).getName());
147         }
148         // If someone else owns the field
149         else {
150             // If affordable rent
151             if (playerlist[currentPlayer].account.getScore() >=
152                     gameboard.getField(playerlist[currentPlayer].getPosition()).getRent(
153                             gameboard.getField(playerlist[currentPlayer].getPosition()).get
    Numberofhouses())){
154                 GUIC.fieldTax(gameboard.getField(playerlist[currentPlayer].getPosition()).g
    etName(),
155                         gameboard.getField(playerlist[currentPlayer].getPosition()).getOwne
    r().getName(),
156                         gameboard.getField(playerlist[currentPlayer].getPosition()).getRent
157                         (gameboard.getField(playerlist[currentPlayer].getPosition()).getOwn
    er().getFleets()-1));
158                 gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
    yerlist[currentPlayer]);
159             // if insufficient funds
160             } else {
161                 GUIC.insufficiantFunds(gameboard.getField(playerlist[currentPlayer].getPosi
    tion()).getName(),
162                         gameboard.getField(playerlist[currentPlayer].getPosition()).getOwne
    r().getName(),
163                         playerlist[currentPlayer].account.getScore());
164                 gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
    yerlist[currentPlayer]);
165             }
166
167             GUIC.updateBalance(gameboard.getField(playerlist[currentPlayer].getPosition()).g
    etOwner().getName(),
168                     gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner().a
    ccount.getScore());
169             GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
170         }
171     }
172
173     public void landOnLaborCamp(int currentPlayer) {
174         // If no owner
175         if (gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner() == null)
    {
176             // If affordable
177             if (playerlist[currentPlayer].account.getScore() >=
    gameboard.getField(playerlist[currentPlayer].getPosition()).getPrice()) {
178                 // Player chooses if he wants to buy the field
179                 boolean buyfield =
    GUIC.buyField(gameboard.getField(playerlist[currentPlayer].getPosition()).getName(),
180                         gameboard.getField(playerlist[currentPlayer].getPosition()).getPric
```

```
e());
                gameboard.getField(playerlist[currentPlayer].getPosition()).setBuyfield(buy
field);
                gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
yerlist[currentPlayer]);

                // Field status on GUI
                if (buyfield) {
                    GUIC.fieldBought(gameboard.getField(playerlist[currentPlayer].getPositi
on()).getName());
                    GUIC.setOwner(playerlist[currentPlayer]);
                    GUIC.updateBalance(playerlist[currentPlayer].getName(),
playerlist[currentPlayer].account.getScore());
                } else
                    GUIC.fieldRefused(gameboard.getField(playerlist[currentPlayer].getPosit
ion()).getName());
                } else
                    GUIC.fieldRefusedPrice(gameboard.getField(playerlist[currentPlayer].get
Position()).getName());
        }
        // If its your own field
        else if (gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner() ==
playerlist[currentPlayer]) {
            GUIC.fieldOwnedByPlayer(gameboard.getField(playerlist[currentPlayer].getPositio
n()).getName());
        }
        // If someone else owns the field
        else {
            // If affordable rent
            if (playerlist[currentPlayer].account.getScore() >=
                    gameboard.getField(playerlist[currentPlayer].getPosition()).getRent(
                            gameboard.getField(playerlist[currentPlayer].getPosition()).get
Owner().getLaborCamp())){
                GUIC.fieldTax(gameboard.getField(playerlist[currentPlayer].getPosition()).g
etName(),
                        gameboard.getField(playerlist[currentPlayer].getPosition()).getOwne
r().getName(),
                        gameboard.getField(playerlist[currentPlayer].getPosition()).getRent
                        (gameboard.getField(playerlist[currentPlayer].getPosition()).getOwn
er().getLaborCamp()));
                gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
yerlist[currentPlayer]);
            // if insufficient funds
            } else {
                GUIC.insufficiantFunds(gameboard.getField(playerlist[currentPlayer].getPosi
tion()).getName(),
                        gameboard.getField(playerlist[currentPlayer].getPosition()).getOwne
r().getName(),
                        playerlist[currentPlayer].account.getScore());
                gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
yerlist[currentPlayer]);
            }

            GUIC.updateBalance(gameboard.getField(playerlist[currentPlayer].getPosition()).g
etOwner().getName(),
                    gameboard.getField(playerlist[currentPlayer].getPosition()).getOwner().a
ccount.getScore());
            GUIC.updateBalance(playerlist[currentPlayer].getName(),
playerlist[currentPlayer].account.getScore());
        }
    }
    public void landOnTax(int currentPlayer) {
```

```
223                 if (gameboard.getField(playerlist[currentPlayer].getPosition()).isOption())
    {
224                     boolean paypercent =
    GUIC.taxPick(gameboard.getField(playerlist[currentPlayer].getPosition()).getName());
225                     if(paypercent) {
226                         gameboard.getField(playerlist[currentPlayer].getPosition()).setPayp
    ercent(true);
227                         gameboard.getField(playerlist[currentPlayer].getPosition()).setNetw
    orth(checkNetworth(currentPlayer));
228                         GUIC.messageTax10percent();
229                     } else if (playerlist[currentPlayer].account.getScore() >=
230                             gameboard.getField(playerlist[currentPlayer].getPosition()).get
    Price()) {
231                         gameboard.getField(playerlist[currentPlayer].getPosition()).setPayp
    ercent(false);
232                         GUIC.taxMessageNoOption(gameboard.getField(playerlist[currentPlayer
    .getPosition()).getPrice());
233                     } else
234                         GUIC.insufficiantFundsTax();
235                 } else {
236                         if (playerlist[currentPlayer].account.getScore() >=
237                             gameboard.getField(playerlist[currentPlayer].getPosition()).get
    Price()) {
238                             GUIC.taxMessageNoOption(gameboard.getField(playerlist[currentPl
    ayer].getPosition()).getPrice());
239             }           else
240                         GUIC.insufficiantFundsTax();
241                 }
242                 gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(pla
    yerlist[currentPlayer]);
243                 GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
244                 gameboard.getField(playerlist[currentPlayer].getPosition()).setNetworth(0);
245             }
246     public void landOnRefuge(int currentPlayer) {
247         GUIC.bonusMessage(playerlist[currentPlayer].getName(),
    gameboard.getField(playerlist[currentPlayer].getPosition()).getRent(0));
248         gameboard.getField(playerlist[currentPlayer].getPosition()).landOnField(playerlist[c
    urrentPlayer]);
249         GUIC.updateBalance(playerlist[currentPlayer].getName(),
    playerlist[currentPlayer].account.getScore());
250     }
251
252     public int checkNetworth(int currentPlayer) {
253         int ownableworth = 0;
254         int buildingworth = 0;
255         for (int i=0; i<40;i++) {
256             if (playerlist[currentPlayer].equals(gameboard.getField(i).getOwner())) {
257                 ownableworth += gameboard.getField(i).getPrice();
258                 buildingworth += (gameboard.getField(i).getNumberofhouses() *
    gameboard.getField(i).getHouseprice());
259             }
260         }
261         int networth = playerlist[currentPlayer].account.getScore() + ownableworth +
    buildingworth;
262         return networth;
263     }
264
265 }
266
```

```java
package control;
import boundary.GUIController;
import deck.Card;
import deck.Deck;
import deck.MoveCard;
import entity.Player;
import fields.GameBoard;

public class DeckController {
    private GUIController GUIC = new GUIController();
    private Deck deck;
    private Player[] playerlist;
    private FieldController FC;
    private int decklength = 30;
    private int cardsdrawned= 0;



    public DeckController(GUIController GUIC, Player[] playerlist, GameBoard board, FieldController FC) {
        this.GUIC = GUIC;
        this.playerlist = playerlist;
        this.deck = new Deck(playerlist, board);
        this.FC = FC;
        shuffleDeck();
    }

    public void drawCard(int currentPlayer) {
        GUIC.showMessage(deck.getMessage(0));
        Card temp = deck.drawCard(playerlist[currentPlayer]);
        cardsdrawned++;
        GUIC.newPosition(playerlist[currentPlayer]);
        if (cardsdrawned >=decklength) {
            deck.shuffleDeck();
            cardsdrawned = 0;
        }
        if (temp instanceof MoveCard)
            FC.landOnField(currentPlayer);
        // ektra update on GUI
        GUIC.updateBalance(playerlist[currentPlayer].getName(), playerlist[currentPlayer].account.getScore());
    }
    public void shuffleDeck() {
        deck.shuffleDeck();
    }
    public void pickCard(Player player, int cardnumber) {
        GUIC.showMessage(deck.getMessage(cardnumber));
        deck.pickCard(player, cardnumber);
        GUIC.newPosition(player);
    }


}

```

```java
package entity;

public class Account {
    private int balance;

    // Object that stores the score of the two players
    public Account() {
        balance = 30000;
    }

    // Public get method to get the score of chosen player
    public int getScore() {
        return balance;
    }

    // Public set method to set the score of a chosen player
    public void setScore(int balance){
        this.balance = balance;
    }

    // Public method to add points to chosen player
    public boolean addPoints(int points) {

        if (balance + points >= 0) {
            balance += points;
            return true;
        } else {
            balance = 0;
            return false;
        }
    }
}

```

```java

package entity;

public class Player {
    private String name;
    private int position, fleets, laborCamp,
    fieldammount_blue,fieldammount_pink,fieldammount_green,fieldammount_grey,
    fieldammount_red,fieldammount_white,fieldammount_yellow,fieldammount_magenta,
    houseammount,hotelammount, jailcount;
    private boolean lost, outofjailcard,jailed,build_blue, build_pink, build_green,
            build_grey, build_red, build_white, build_yellow, build_magenta,
  paydoublefleet;
    public Account account = new Account();

    // Object that stores the name and position of a player
    public Player(String name) {
        position = 0;
        fleets = 0;
        laborCamp = 0;
        lost = false;
        outofjailcard = false;
        jailed = false;
        jailcount = 0;
        fieldammount_blue = 0;
        fieldammount_pink = 0;
        fieldammount_grey = 0;
        fieldammount_green = 0;
        fieldammount_red = 0;
        fieldammount_white = 0;
        fieldammount_yellow = 0;
        fieldammount_magenta = 0;
        houseammount = 0;
        hotelammount = 0;
        this.name = name;
        build_blue = false;
        build_pink = false;
        build_green = false;
        build_grey = false;
        build_red = false;
        build_white = false;
        build_yellow = false;
        build_magenta = false;
        paydoublefleet = false;
    }

    // Method that returns the name of the player
    public String getName() {
        return name;
    }

    // Method that sets the position of the player
    public void movePosition(int move) {
        if (position + move >= 40) {
            position = position + move - 40;
            passStart();
        } else {
            position += move;
        }
    }
    public void passStart(){
        account.addPoints(4000);
    }
```

```
62      public void setPosition(int position){
63          this.position=position;
64      }
65      // Method that returns the position of the player
66      public int getPosition() {
67          return position;
68      }
69      // Adds a fleet to the players owned fleets
70      public void setFleets(int fleets) {
71          this.fleets = fleets;
72      }
73
74      public void addFleet(){
75          fleets++;
76      }
77
78      // Returns the number of the players owned fleets
79      public int getFleets() {
80          return fleets;
81      }
82      public void addLaborCamp(){
83          laborCamp++;
84      }
85
86      public void setLaborCamp(int laborcamp){
87          this.laborCamp = laborcamp;
88      }
89
90      public int getLaborCamp(){
91          return laborCamp;
92      }
93      public void setStatus(boolean lost) {
94          this.lost = lost;
95      }
96      public boolean getStatus() {
97          return lost;
98      }
99
100     // Getters, setters and add for Fieldammount below.
101     public int getFieldammount_blue() {
102         return fieldammount_blue;
103     }
104
105     public void setFieldammount_blue(int fieldammount_blue) {
106         this.fieldammount_blue = fieldammount_blue;
107     }
108
109     public void addFieldammount_blue(){
110         fieldammount_blue++;
111     }
112
113     public int getFieldammount_pink() {
114         return fieldammount_pink;
115     }
116
117     public void setFieldammount_pink(int fieldammount_pink) {
118         this.fieldammount_pink = fieldammount_pink;
119     }
120
121     public void addFieldammount_pink(){
122         fieldammount_pink++;
123     }
```

```
124
125     public int getFieldammount_green() {
126         return fieldammount_green;
127     }
128
129     public void setFieldammount_green(int fieldammount_green) {
130         this.fieldammount_green = fieldammount_green;
131     }
132
133     public void addFieldammount_green(){
134         fieldammount_green++;
135     }
136
137     public int getFieldammount_grey() {
138         return fieldammount_grey;
139     }
140
141     public void setFieldammount_grey(int fieldammount_grey) {
142         this.fieldammount_grey = fieldammount_grey;
143     }
144
145     public void addFieldammount_grey(){
146         fieldammount_grey++;
147     }
148
149     public int getFieldammount_red() {
150         return fieldammount_red;
151     }
152
153     public void setFieldammount_red(int fieldammount_red) {
154         this.fieldammount_red = fieldammount_red;
155     }
156
157     public void addFieldammount_red(){
158         fieldammount_red++;
159     }
160
161     public int getFieldammount_white() {
162         return fieldammount_white;
163     }
164
165     public void setFieldammount_white(int fieldammount_white) {
166         this.fieldammount_white = fieldammount_white;
167     }
168
169     public void addFieldammount_white(){
170         fieldammount_white++;
171     }
172
173     public int getFieldammount_yellow() {
174         return fieldammount_yellow;
175     }
176
177     public void setFieldammount_yellow(int fieldammount_yellow) {
178         this.fieldammount_yellow = fieldammount_yellow;
179     }
180
181     public void addFieldammount_yellow(){
182         fieldammount_yellow++;
183     }
184
185     public int getFieldammount_magenta() {
```

```
186         return fieldammount_magenta;
187     }
188 
189     public void setFieldammount_magenta(int fieldammount_magenta) {
190         this.fieldammount_magenta = fieldammount_magenta;
191     }
192 
193     public void addFieldammount_magenta(){
194         fieldammount_magenta++;
195     }
196 
197     //getters and setters for buildings
198 
199     public boolean getBuy_Blue(){
200         return build_blue;
201     }
202 
203     public void setBuy_Blue(boolean khan){
204         this.build_blue = khan;
205     }
206 
207     public boolean getBuy_Pink(){
208         return build_pink;
209     }
210 
211     public void setBuy_Pink(boolean khan){
212         this.build_pink = khan;
213     }
214 
215     public boolean getBuy_Green(){
216         return build_green;
217     }
218 
219     public void setBuy_Green(boolean khan){
220         this.build_green = khan;
221     }
222 
223     public boolean getBuy_grey(){
224         return build_grey;
225     }
226 
227     public void setBuy_grey(boolean khan){
228         this.build_grey = khan;
229     }
230 
231     public boolean getBuy_Red(){
232         return build_red;
233     }
234 
235     public void setBuy_Red(boolean khan){
236         this.build_red = khan;
237     }
238 
239     public boolean getBuy_White(){
240         return build_white;
241     }
242 
243     public void setBuy_White(boolean khan){
244         this.build_white = khan;
245     }
246 
247     public boolean getBuy_Yellow(){
```

```
248         return build_yellow;
249     }
250 
251     public void setBuy_Yellow(boolean khan){
252         this.build_yellow = khan;
253     }
254 
255     public boolean getBuy_Magenta(){
256         return build_magenta;
257     }
258 
259     public void setBuy_Magenta(boolean khan){
260         this.build_magenta = khan;
261     }
262     // Houseammount and hotelammount listed below
263 
264     public int getHouseammount() {
265         return houseammount;
266     }
267 
268     public void setHouseammount(int houseammount) {
269         this.houseammount = houseammount;
270     }
271 
272     public void addHouseammount(int houseAmount){
273         houseammount += houseAmount;
274     }
275 
276     public int getHotelammount() {
277         return hotelammount;
278     }
279 
280     public void setHotelammount(int hotelammount) {
281         this.hotelammount = hotelammount;
282     }
283 
284     public void addHotelammount(int hotelAmount){
285         hotelammount+= hotelAmount;
286     }
287 
288     // Jailcard and jail status
289 
290     public void setOutofjailcard(boolean outofjailcard) {
291         this.outofjailcard = outofjailcard;
292     }
293 
294     public void removeOutofjailcard(){
295         outofjailcard = false;
296     }
297 
298     public boolean isJailed() {
299         return jailed;
300     }
301 
302     public void setJailed(boolean jailed) {
303         this.jailed = jailed;
304     }
305 
306     public boolean hasOutofjailcard() {
307         return outofjailcard;
308     }
309 
```

```
310     public int getJailcount() {
311         return jailcount;
312     }
313 
314     public void setJailcount(int jailcount) {
315         this.jailcount = jailcount;
316     }
317     public void addJailcount() {
318         jailcount++;
319     }
320 
321     public String getOwner() {
322         return null;
323     }
324 
325     public boolean isPaydoublefleet() {
326         return paydoublefleet;
327     }
328 
329     public void setPaydoublefleet(boolean paydoublefleet) {
330         this.paydoublefleet = paydoublefleet;
331     }
332 
333 }
334 
```

```java

package entity;

public class Dice {

    private int dice;
    private int diceside = 6;

    public Dice(){
        super();
    }

    public int getDice(){
        return dice;
    }

    public void setDice(int dice){
        this.dice = dice;
    }

    public int getDiceside(){
        return diceside;
    }

    public void setDiceside(int new_diceside) {
        this.diceside = new_diceside;
    }

    public int rollDice(){
        dice = (int)(Math.random()*diceside+1);
        return dice;
    }

    public String toString() {
        return ("The dice with " + diceside + " sides, has the current value of: " + dice);
    }

}

```

```java
package entity;


    public class DiceBox {

        //Dice t1;
        //Dice t2;
        int numberOfDice = 2;
        Dice[] box  = new Dice[numberOfDice];

        public DiceBox(){
        this.box[0] = new Dice();
        this.box[1] = new Dice();
        }

        public int getDice(int dicenumber) {
            return box[dicenumber].getDice();
        }

        public int getDice1(){
            return box[0].getDice();
        }

        public int getDice2(){
            return box[1].getDice();
        }

        public void setDice(int dicenumber, int new_dicevalue) {
            box[dicenumber].setDice(new_dicevalue);
        }

        public int getDiceside(int dicenumber) {
            return box[dicenumber].getDiceside();
        }

        public void rollDice(){
            box[0].rollDice();
            box[1].rollDice();
        }

        // Tjek for par
        public boolean isEqual(){
            if (box[0].getDice() == box[1].getDice())
                return true;
            else
                return false;
        }

        public int getSum() {
            return (box[0].getDice() + box[1].getDice());
        }

        public void setDiceside(int dicenumber, int new_diceside){
            box[dicenumber].setDiceside(new_diceside);
        }

        public String toString() {
            return ("The dicebox has the values " + box[0].getDice() + " and " +
  box[1].getDice());
        }
    }

```

62
63

```java

package fields;

import entity.Player;

public abstract class Field {
    protected String name;

    public Field(String name) {
        this.name = name;
    }
    public abstract void landOnField(Player player);

    public abstract String toString();

    public Player getOwner() {
        return null;
    }
    public int getPrice() {
        return 0;
    }
    public int getHouseprice() {
        return 0;
    }
    public int getNumberofhouses() {
        return 0;
    }

    public void setNumberofHouses(int numberofhouses){

    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public boolean isBuyfield() {
        return false;
    }
    public void setBuyfield(boolean buyfield) {

    }
    public int getRent(int rentnumber) {
        return 0;
    }
    public boolean isOption() {
        return false;
    }
    public boolean isPaypercent() {
        return false;
    }
    public void setPaypercent(boolean paypercent) {
        }
    public boolean checkPayDoubleRent(Player player){
        return false;
    }
    public void addNumberofHouses(int add){
    }
    public int getNetworth(){
        return 0;
    }
```

```
63     public void setNetworth(int networth){
64
65     }
66
67 }
68
```

```java
package fields;

import entity.DiceBox;
import entity.Player;

public class GameBoard {

    private Field[] fieldlist;

    public GameBoard(DiceBox box) {

        // Array that creates each field and the attributes
        fieldlist = new Field[40];
        fieldlist[0] = new Start("Start");
        fieldlist[1] = new Territory("Rødovrevej",1200, 1000, 600, 40, 200, 600, 1800, 3200, 5000,"Blue");
        fieldlist[2] = new ChanceCard("Chancekort");
        fieldlist[3] = new Territory("Hvidovrevej", 1200, 1000, 600, 80, 400, 1200, 3600, 6400, 9000,"Blue");
        fieldlist[4] = new Tax("Statsskat", 4000, true);
        fieldlist[5] = new Fleet("Helsingør-Helsingborg", 4000, 2000, 500, 1000, 2000, 4000);
        fieldlist[6] = new Territory("Roskildevej", 2000, 1000, 1000, 120, 600, 1800, 5400, 8000, 11000,"Pink");
        fieldlist[7] = new ChanceCard("Chancekort");
        fieldlist[8] = new Territory("Valby Langgade", 2000, 1000, 1000, 120, 600, 1800, 5400, 8000, 11000, "Pink");
        fieldlist[9] = new Territory("Allegade", 2400, 1000, 1200, 160, 800, 2000, 6000, 9000, 12000, "Pink");
        fieldlist[10] = new Jail("Fængsel");
        fieldlist[11] = new Territory("Fredriksberg Alle", 2800, 2000, 1400, 200, 1000, 3000, 9000, 12500, 15000, "Green");
        fieldlist[12] = new LaborCamp("Tuborg", 3000, 1500, 80, box);
        fieldlist[13] = new Territory("Bulowsvej", 2800, 2000, 1400, 200, 1000, 3000, 9000, 12500, 15000,"Green");
        fieldlist[14] = new Territory("Gl Kongevej", 3200, 2000, 1600, 240, 1200, 3600, 10000, 14000, 18000, "Green");
        fieldlist[15] = new Fleet("Mols-Linien", 4000, 2000, 500, 1000, 2000, 4000);
        fieldlist[16] = new Territory("Bernstorffsvej", 3600, 2000, 1800, 280, 1400, 4000, 11000, 15000, 19000,"Gray");
        fieldlist[17] = new ChanceCard("Chancekort");
        fieldlist[18] = new Territory("Hellerupvej", 3600, 2000, 1800, 280, 1400, 4000, 11000, 15000, 19000, "Gray");
        fieldlist[19] = new Territory("Strandvejen", 4000, 2000, 2000, 320, 1600, 4400, 12000, 16000, 20000, "Gray");
        fieldlist[20] = new Refuge("Parkering", 5000);
        fieldlist[21] = new Territory("Trianglen", 4400, 3000, 2200, 360, 1800, 5000, 14000, 17500, 21000,"Red");
        fieldlist[22] = new ChanceCard("Chancekort");
        fieldlist[23] = new Territory("Østerbrogade", 4400, 3000, 2200, 360, 1800, 5000, 14000, 17500, 21000,"Red");
        fieldlist[24] = new Territory("Grønningen", 4800, 3000, 2400, 400, 2000, 6000, 15000, 18500, 22000,"Red");
        fieldlist[25] = new Fleet("Gedser-Rostock", 4000, 2000, 500, 1000, 2000, 4000);
        fieldlist[26] = new Territory("Bredgade", 5200, 3000, 2600, 440, 2200, 6600, 16000, 19500, 23000,"White");
        fieldlist[27] = new Territory("Kgs Nytorv", 5200, 3000, 2600, 440, 2200, 6600, 16000, 19500, 23000,"White");
        fieldlist[28] = new LaborCamp("Coca-Cola", 3000, 1500, 80, box);
        fieldlist[29] = new Territory("Østergade", 5600, 3000, 2800, 480, 2400, 7200, 17000, 20500, 24000,"White");
        fieldlist[30] = new MoveToJail("Gå I Fængslet");
```

```
45          fieldlist[31] = new Territory("Amagertorv", 6000, 4000, 3000, 520, 2600, 7800,
  18000, 22000, 25500,"Yellow");
46          fieldlist[32] = new Territory("Vimmelskaftet", 6000, 4000, 3000, 520, 2600, 7800,
  18000, 22000, 25500,"Yellow");
47          fieldlist[33] = new ChanceCard("Chancekort");
48          fieldlist[34] = new Territory("Nygade", 6400, 4000, 3200, 560, 3000, 9000, 20000,
  24000, 28000,"Yellow");
49          fieldlist[35] = new Fleet("Rødby-Puttgarden", 4000, 2000, 500, 1000, 2000, 4000);
50          fieldlist[36] = new ChanceCard("Chancekort");
51          fieldlist[37] = new Territory("Frederiksberggade", 7000, 4000, 3500, 700, 3500,
  10000, 22000, 26000, 30000, "Magneta");
52          fieldlist[38] = new Tax("Indkomstskat", 2000, false);
53          fieldlist[39] = new Territory("Rådhuspladsen", 8000, 4000, 4000, 1000, 4000, 12000,
  28000, 34000, 40000, "Magneta");
54
55
56
57
58      }
59
60      public Field getField(int i) {
61          return fieldlist[i];
62      }
63      public void resetOwnedFields(Player player) {
64          for (int i = 0; i<fieldlist.length; i++) {
65              if (fieldlist[i] instanceof Ownable) {
66                  if (((Ownable) fieldlist[i]).getOwner() == player) {
67                      ((Ownable) fieldlist[i]).setOwner(null);
68                  }
69              }
70          }
71      }
72      public String toString() {
73          String s = "";
74          for (int i = 0; i<fieldlist.length; i++) {
75              s += "Felt " + (i+1)+ "     " + fieldlist[i].toString();
76          }
77          return s;
78      }
79  }
80
```

```java
package fields;

import entity.Player;

public abstract class Ownable extends Field {

    protected Player owner;
    protected int price;
    protected int pansat;
    protected boolean buyfield = false;

    public Ownable(String name, int price, int pansat) {
        super(name);
        this.price = price;
        this.pansat = pansat;
    }

    @Override
    public abstract void landOnField(Player player);

    @Override
    public abstract String toString();

    public void setOwner (Player player) {
        owner = player;
    }
    public Player getOwner() {
        return owner;
    }
    public int getPrice() {
        return price;
    }

    @Override
    public boolean isBuyfield() {
        return buyfield;
    }

    @Override
    public void setBuyfield(boolean buyfield) {
        this.buyfield = buyfield;
    }
}

```

```java
package fields;

import entity.Player;

public class Fleet extends Ownable {
    private int[] rent = new int[4];

    public Fleet(String name, int price, int pansat, int rent_1, int rent_2, int rent_3,
            int rent_4) {
        super(name, price, pansat);
        this.rent[0] = rent_1;
        this.rent[1] = rent_2;
        this.rent[2] = rent_3;
        this.rent[3] = rent_4;

    }
    @Override
    public int getRent(int numberoffleets) {
        return rent[numberoffleets];
    }

    @Override
    public void landOnField(Player player) {
        // If the current field has no owner, the player can buy it
        if (getOwner() == null) {
            if (player.account.getScore() >= price) {
                if (buyfield) {
                    player.account.addPoints(-price);
                    setOwner(player);
                    player.addFleet();
                }       }
            // if the owner is the player himself, nothing happens
        } else if (getOwner().equals(player)) {
        }
        // if the field is owned by another player, a rent have to be paid
        else if (player.isPaydoublefleet()){
            if (player.account.getScore() >= rent[getOwner().getFleets() - 1]*2) {
                player.account.addPoints(-rent[getOwner().getFleets() - 1]*2);
                getOwner().account.addPoints(rent[getOwner().getFleets() - 1]*2);
                player.setPaydoublefleet(false);
                // the player loses if the rent is higher than the players
                // balance
            } else {
                getOwner().account.addPoints(player.account.getScore());
                player.account.addPoints(-player.account.getScore());
                player.setStatus(true);
            }
        }
        else {
            if (player.account.getScore() >= rent[getOwner().getFleets() - 1]) {
                player.account.addPoints(-rent[getOwner().getFleets() - 1]);
                getOwner().account.addPoints(rent[getOwner().getFleets() - 1]);
                // the player loses if the rent is higher than the players
                // balance
            } else {
                getOwner().account.addPoints(player.account.getScore());
                player.account.addPoints(-player.account.getScore());
                player.setStatus(true);
            }
        }
    }

```

```
63      @Override
64      public String toString() {
65          return "Type: Fleet --- Name: " + name + " --- Price: " + price + " --- Rent 1: " +
  rent[0]
66                  + " --- Rent 2: " + rent[1] + " --- Rent 3: " + rent[2] + " --- Rent 4: "
67                  + rent[3] + "\n";
68      }
69  }
70
```

```java
package fields;

import entity.DiceBox;
import entity.Player;


public class LaborCamp extends Ownable {

    private int rent;
    private DiceBox box;
    private int fullRent;

    public LaborCamp(String name, int price, int pansat, int rent, DiceBox box) {
        super(name, price, pansat);
        this.rent = rent;
        this.box = box;
    }
    @Override
    public int getRent(int numberoflaborcamps) {
        if (numberoflaborcamps == 1)
            return 80*box.getSum();
        else if (numberoflaborcamps==2)
            return 200*box.getSum();
        else
            return 0;
    }

    @Override
    public void landOnField(Player player) {
        // If the current field has no owner, the player can buy it
        if (getOwner() == null) {
            if (player.account.getScore() >= price) {
                if (buyfield) {
                    player.account.addPoints(-price);
                    setOwner(player);
                    player.addLaborCamp();
                }
            }
            // if the owner is the player himself, nothing happens
        } else if (getOwner().equals(player)) {
            // nothing happens
        }
        // if the field is owned by another player, a rent have to be paid
        else {
            if(getOwner().getLaborCamp()==2){
                rent = 200;
            }else{
                rent = 80;
            }
            fullRent = rent * box.getSum();
            if (player.account.getScore() >= fullRent) {
                getOwner().account.addPoints(fullRent);
                player.account.addPoints(-fullRent);
            // the player looses if the rent is higher than the players balance
            } else {
                getOwner().account.addPoints(player.account.getScore());
                player.account.addPoints(-player.account.getScore());
                player.setStatus(true);
            }
        }
    }

```

```
63      @Override
64      public String toString() {
65          return "Type: Labor Camp --- Name: " + name + " --- Price: " + price + " --- Rent: "
  + rent + "\n";
66      }
67  }
68
```

```java
package fields;

import entity.Player;

public class Territory extends Ownable {

    private int houseprice, numberofhouses;
    private int[] rent = new int[6];
    private String color;

    public Territory(String name, int price, int houseprice, int pansat,
            int rent1, int rent2, int rent3, int rent4, int rent5, int hotel, String color)
  {
        super(name, price, pansat);
        rent[0] = rent1;
        rent[1] = rent2;
        rent[2] = rent3;
        rent[3] = rent4;
        rent[4] = rent5;
        rent[5] = hotel;
        this.houseprice = houseprice;
        numberofhouses = 0;
        this.color = color;
        this.name = name;
        this.price  = price;
        this.pansat = pansat;

    }

    @Override
    public void landOnField(Player player) {
        // If the current field has no owner, the player can buy it
        if (getOwner() == null) {
            if (player.account.getScore() >= price) {
                if (buyfield) {
                    player.account.addPoints(-price);
                    setOwner(player);
                    buyfield = false;
                    switch(color){
                    case "Blue"    : player.addFieldammount_blue();
                    break;
                    case "Pink"    : player.addFieldammount_pink();
                    break;
                    case "Green"   : player.addFieldammount_green();
                    break;
                    case "Gray"    : player.addFieldammount_grey();
                    break;
                    case "Red"     : player.addFieldammount_red();
                    break;
                    case "White"   : player.addFieldammount_white();
                    break;
                    case "Yellow" : player.addFieldammount_yellow();
                    break;
                    case "Magneta": player.addFieldammount_magenta();
                    break;
                    }
                }
            }
        // if the owner is the player himself, nothing happens
        } else if (getOwner().equals(player)) {
            // Nothing happens
        // if the field is owned by another player, a rent have to be paid
```

```
62          } else {
63              if (checkPayDoubleRent(player)){
64                  if(player.account.getScore() >= rent[0]*2)
65                      payDoubleRent(player);
66                  // the player loses if the rent is higher than the players balance
67                  else {
68                      getOwner().account.addPoints(player.account.getScore());
69                      player.account.addPoints(-player.account.getScore());
70                      player.setStatus(true);
71                  }
72              }
73              else if(player.account.getScore() >= rent[numberofhouses])
74                      payRent(player);
75              // the player loses if the rent is higher than the players balance
76              else {
77                  getOwner().account.addPoints(player.account.getScore());
78                  player.account.addPoints(-player.account.getScore());
79                  player.setStatus(true);
80              }
81          }
82      }
83
84      public void payRent(Player player){
85          getOwner().account.addPoints(rent[numberofhouses]);
86          player.account.addPoints(-rent[numberofhouses]);
87      }
88      @Override
89      public boolean checkPayDoubleRent(Player player){
90          switch(color){
91              case "Blue" :
92                  if (getOwner().getFieldammount_blue()==2 && numberofhouses == 0)
93                      return true;
94                  else
95                      return false;
96              case "Pink" :
97                  if (getOwner().getFieldammount_pink()==3 && numberofhouses == 0)
98                      return true;
99                  else
100                     return false;
101             case "Green" :
102                 if (getOwner().getFieldammount_green()==3 && numberofhouses == 0)
103                     return true;
104                 else
105                     return false;
106             case "Gray" :
107                 if (getOwner().getFieldammount_grey()==3 && numberofhouses == 0)
108                     return true;
109                 else
110                     return false;
111             case "Red" :
112                 if (getOwner().getFieldammount_red()==3 && numberofhouses == 0)
113                     return true;
114                 else
115                     return false;
116             case "White" :
117                 if (getOwner().getFieldammount_white()==3 && numberofhouses == 0)
118                     return true;
119                 else
120                     return false;
121             case "Yellow" :
122                 if (getOwner().getFieldammount_yellow()==3 && numberofhouses == 0)
123                     return true;
```

```
124                 else
125                     return false;
126             case "Magneta" :
127                 if (getOwner().getFieldammount_magenta()==2 && numberofhouses == 0)
128                     return true;
129                 else
130                     return false;
131             default:
132                     return false;
133 
134         }
135     }
136 
137 
138     public void payDoubleRent(Player player){
139         getOwner().account.addPoints(rent[0]*2);
140         player.account.addPoints(-rent[0]*2);
141     }
142 
143     public int getHouseprice() {
144         return houseprice;
145     }
146     public int getNumberofhouses() {
147         return numberofhouses;
148     }
149     public void setNumberofHouses(int numberofhouses){
150         this.numberofhouses += numberofhouses;
151     }
152     public void addNumberofHouses(int add){
153         numberofhouses += add;
154     }
155     public String getColor(){
156         return color;
157     }
158 
159     @Override
160     public String toString() {
161         return "Type: Territory --- Name: " + name + " --- Price: " + price + " --- Rent: "
    + rent + "\n";
162     }
163 
164     public int getRent(int numberofhouses) {
165         return rent[numberofhouses];
166     }
167 
168 }
169 
```

```java
package fields;

import entity.Player;

public class Jail extends Field{

    public Jail(String name){
        super(name);
    }

    @Override
     public void landOnField(Player player){
     }

    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return null;
    }


}

```

```java
package fields;

import entity.Player;


public class Refuge extends Field {

private int bonus;

    public Refuge(String name, int bonus) {
        super(name);
        this.bonus = bonus;
        this.name = name;
    }
    @Override
    public void landOnField(Player player) {
            player.account.addPoints(bonus);
    }
    @Override
    public int getRent(int t) {
        return bonus;
    }
    @Override
    public String toString() {
        return "Type: Refuge --- Name: " + name + " --- Bonus: " + bonus + "\n";
    }

}

```

```java
package fields;

import entity.Player;

public class Start extends Field {

    public Start(String name) {
        super(name);
    }

    @Override
    public void landOnField(Player player) {
    }
    @Override
    public String toString() {
        return "Type: Start --- Name: " + name + "\n";
    }

}

```

```java
package fields;

import entity.Player;

public class Tax extends Field {

    private String name;
    private int pay;
    private boolean option;
    private boolean paypercent=false;
    private int networth;

    public Tax(String name, int pay, boolean option) {
        super(name);
        this.pay = pay;
        this.name = name;
        this.option = option;
    }

    @Override
    public void landOnField(Player player) {
        if (option) {
            if (paypercent)
            player.account.addPoints(-(networth / 10));
            else if (player.account.getScore() >= pay)
                player.account.addPoints(-pay);
            else {
                player.account.addPoints(-player.account.getScore());
                player.setStatus(true);
            }
        }
        else {
            if (player.account.getScore() >= pay) {
                    player.account.addPoints(-pay);
            } else {
                player.account.addPoints(-player.account.getScore());
                player.setStatus(true);
            }

        }
    }
    public boolean isPaypercent() {
        return paypercent;
    }
    public void setPaypercent(boolean paypercent) {
            this.paypercent = paypercent;
    }
    @Override
    public String toString() {
        return "Type: Tax --- Name: " + name + " --- Tax: " + pay + "\n";
    }
    public boolean isOption() {
        return option;
    }
    public void setOption(boolean option) {
        this.option = option;
    }
    @Override
    public int getPrice() {
        return pay;
    }
    @Override
```

```
63      public int getNetworth() {
64          return networth;
65      }
66      @Override
67      public void setNetworth(int networth) {
68          this.networth = networth;
69      }
70
71  }
72
```

```java
package deck;
import entity.Player;
public abstract class Card {

    private String message;

    public Card(String message) {
        this.message = message;
    }

    public abstract void drawCard(Player player);

    public String getMessage() {
        return message;
    }

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

}

```

```java
package deck;
import entity.Player;
import fields.GameBoard;
import java.util.Random;

public class Deck {
    private Card[] cardlist = new Card[30];

    public Deck(Player[] playerlist, GameBoard board){
        cardlist[0] = new MoveToCard("Ryk frem til Grønningen. Hvis de passerer \"Start\", indkasser da 4000kr ", 24, board);
        cardlist[1] = new MoveToFleetCard("Ryk brikken frem til det nærmeste rederi og betal ejeren to gange den leje, "
                + "han ellers er berettiget til, Hvis selskabet ikke ejes af nogen kan De købe det af banken.", board);
        cardlist[2] = new MoveToCard("Ryk frem til \"Start\"", 0, board);
        cardlist[3] = new MoveAmmountCard("Ryk tre felter tilbage", -3, board);
        cardlist[4] = new MoveToCard("Ryk frem til Frederiksberg Alle. Hvis de passerer \"Start\", indkasser da 4000kr", 11, board);
        cardlist[5] = new MoveToCard("Tag med Mols-Linjen -- Flyt brikken frem og hvis de passerer \"Start\", indkasser da 4000kr", 15, board);
        cardlist[6] = new MoveToCard("Tag ind på Rådhuspladsen", 39, board);
        cardlist[7] = new JailCard("Gå i Fængsel. Ryk direkte til faengsel. Selv om De passerer \"Start\", indkassere de ikke 4000kr", board);
        cardlist[8] = new JailCard("Gå i Fængsel. Ryk direkte til fÃ¦ngsel. Selv om De passerer \"Start\", indkassere de ikke 4000kr", board);
        cardlist[9] = new PayCard("Betal 3000kr for reparation af Deres vogn", 3000);
        cardlist[10] = new PropertyPayCard("Oliepriserne er steget, og de skal betale: \n 500kr pr. hus \n 2000kr pr. hotel", 500, 2000);
        cardlist[11] = new PayCard("De har måtte vedtage en parkeringsbøde. Betal 200kr i både", 200);
        cardlist[12] = new PayCard("Betal deres bilforsikring på 1000kr", 1000);
        cardlist[13] = new PayCard("Betal 3000kr for reparation af deres vogn", 3000);
        cardlist[14] = new PayCard("De har været en tur i udlandet og haft for mange cigaretter med hjem. Betal told 200kr", 200);
        cardlist[15] = new PayCard("De har kørt frem for \"Fuld Stop\". Betal 1000kr i bøde", 1000);
        cardlist[16] = new PayCard("De har modtaget Deres tandlægeregning. Betal 2000kr", 2000);
        cardlist[17] = new PropertyPayCard("Ejendomsskatterne er steget, Ekstra udgifter er: \n 800kr pr. hus \n 2300kr pr. hotel", 800, 2300);
        cardlist[18] = new RecieveCard("Værdien af egen avl fra nyttehaven udgoer 200kr, som de modtager af banken", 200);
        cardlist[19] = new GiftCard("Det er Deres fødselsdag. Modtag af hver medspiller 200kr", 200, playerlist);
        cardlist[20] = new RecieveCard("De har vundet i Kasselotteriet. Modtag 500kr", 500);
        cardlist[21] = new RecieveCard("De havde en række med elleve rigtige i tipning. Modtag 1000kr", 1000);
        cardlist[22] = new RecieveCard("Modtag udbytte af Deres aktier 1000kr", 1000);
        cardlist[23] = new RecieveCard("Grundet dyrtiden har De faeet gageforhøjelse. Modtag 1000kr", 1000);
        cardlist[24] = new RecieveCard("Kommunen har eftergivet et kvartals skat. Hæv i banken 3000kr", 3000);
        cardlist[25] = new RecieveCard("Modtag udbytte af Deres aktier 1000kr", 1000);
        cardlist[26] = new RecieveCard("Deres præmieobligation er kommet ud. De modtager 1000kr af banken", 1000);
        cardlist[27] = new RecieveCard("De modtager Deres aktieudbytte. Modtag 1000kr af banken", 1000);
        cardlist[28] = new RecieveIfCard("De modtager \"Matador-legatet for de værdige trængende\" stort 40000kr. "
                + "Ved værdige trængende forstås, af Deres formue, "
                + "d.v.s. Deres kontante penge + skøder + bygninger ikke overskrider
```

```
  15000kr", 40000, 15000, board);
42        cardlist[29] = new QueensCard("I anledning af deres majestæts fødselsdag benådes de
  herved for fængsel. "
43                + "Dette kort kan opbevares, indtil De får brug for det");
44    }
45
46    public void shuffleDeck() {
47        // Fisher-Yates shuffle algorithm
48        Random r = new Random();
49        for (int i = cardlist.length-1; i>0; i--) {
50            int index = r.nextInt(i+1);
51            Card a = cardlist[index];
52            cardlist[index] = cardlist[i];
53            cardlist[i] = a;
54        }
55    }
56    public Card drawCard(Player player) {
57        cardlist[0].drawCard(player);
58        // puts the card in the bottom of the list/array
59        Card temp = cardlist[0];
60        for (int k = 1; k<cardlist.length; k++){
61            cardlist[k-1] = cardlist[k];
62        }
63        cardlist[cardlist.length-1]=temp;
64        return temp;
65
66    }
67    public String getMessage(int cardnumber) {
68        return cardlist[cardnumber].getMessage();
69    }
70    public void pickCard(Player player, int cardnumber) {
71        cardlist[cardnumber].drawCard(player);
72    }
73
74
75    public String toString() {
76        String n = "";
77        for (int i=0; i<cardlist.length; i++){
78            n += i+1 + ":\t" + cardlist[i] + "\n" + "\n";
79        }
80        return n;
81    }
82}
83
```

```java
package fields;

import entity.Player;

public class ChanceCard extends Field{

    public ChanceCard(String name){
        super(name);
    }

    @Override
    public void landOnField(Player player){

    }

    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return null;
    }

}

```

```java
package deck;
import entity.Player;

public class GiftCard extends RecieveCard {
    private Player[] playerlist;

    public GiftCard(String message, int bonus, Player[] playerlist) {
        super(message, bonus);
        this.playerlist = playerlist;
    }

    public void drawCard(Player player) {
        for (int i=0; i < playerlist.length; i++) {
            boolean t = player.equals(playerlist[i]);
            if (!t) {
                if (bonus>playerlist[i].account.getScore()){
                    playerlist[i].setStatus(true);
                    playerlist[i].account.setScore(0);
                }
                else
                playerlist[i].account.addPoints(-bonus);
            }
        }
        player.account.addPoints(bonus*(playerlist.length-1));
    }
}

```

```java
package deck;
import entity.Player;
import fields.GameBoard;

public class JailCard extends MoveCard {
    private int newposition = 10;

    public JailCard(String message, GameBoard board) {
        super(message, board);
    }

    public void drawCard(Player player) {
        player.setPosition(newposition);
        player.setJailed(true);
    }


}

```

```java
package deck;
import entity.Player;
import fields.GameBoard;

public class MoveAmmountCard extends MoveCard{
    private int moveammount;

    public MoveAmmountCard(String message, int moveammount, GameBoard board) {
        super(message, board);
        this.moveammount = moveammount;
    }

    @Override
    public void drawCard(Player player) {
        if (player.getPosition()+moveammount<39)
            player.passStart();
        // control for Chancecard on field 2
        if (player.getPosition() ==2)
            player.setPosition(39);
        else
            player.setPosition(player.getPosition()+moveammount);

    }

}

```

```java
package deck;

import entity.Player;
import fields.GameBoard;

public abstract class MoveCard extends Card{
    protected GameBoard board;

    public MoveCard(String message, GameBoard board) {
        super(message);
        this.board = board;
    }

    public abstract void drawCard(Player player);

}

```

```java
package deck;

import entity.Player;
import fields.GameBoard;
public class MoveToCard extends MoveCard {
    private int newposition;

    public MoveToCard(String message, int newposition, GameBoard board) {
        super(message, board);
        this.newposition = newposition;
    }

    @Override
    public void drawCard(Player player) {
        if (player.getPosition()>=newposition)
            player.passStart();
        player.setPosition(newposition);
    }



}

```

```java
package deck;

import entity.Player;
import fields.GameBoard;

public class MoveToFleetCard extends MoveCard{
    private int newposition;

    public MoveToFleetCard(String message, GameBoard board) {
        super(message, board);
    }

    @Override
    public void drawCard(Player player) {
        if (player.getPosition() >= 35 || player.getPosition() < 5) {
            newposition = 5;
            if (player.getPosition() <= 39)
                player.passStart();
        }
        else if (player.getPosition() >= 25 && player.getPosition() < 35) {
            newposition = 35;
        }
        else if (player.getPosition() >= 15 && player.getPosition() < 25) {
            newposition = 25;
        }
        else if (player.getPosition() >= 5 && player.getPosition() < 15) {
            newposition = 15;
        }
        player.setPaydoublefleet(true);
        player.setPosition(newposition);
    }


}

```

```java
package fields;

import entity.Player;

public class MoveToJail extends Field {

    public MoveToJail(String name) {
        super(name);
    }

    @Override
    public void landOnField(Player player){
        player.setJailed(true);
        player.setPosition(10);

    }

    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return null;
    }

}

```

```java
package deck;
import entity.Player;

public class PayCard extends Card{
    protected int price;

    public PayCard(String message, int price) {
        super(message);
        this.price = price;
    }

    @Override
    public void drawCard(Player player) {
        if (player.account.getScore()<price){
            player.setStatus(true);
            player.account.setScore(0);
        }
        else
        player.account.addPoints(-price);
    }
}

```

```java
package deck;
import entity.Player;

public class PropertyPayCard extends PayCard {
    private int price_house;
    private int price_hotel;

    public PropertyPayCard(String message, int price_house, int price_hotel) {
        super(message, 0);
        this.price_house = price_house;
        this.price_hotel = price_hotel;
    }

    public void drawCard(Player player) {
        if (player.account.getScore()<price_house*player.getHouseammount()+price_hotel*player.getHotelammount()){
            player.setStatus(true);
            player.account.setScore(0);
        }
        else
        player.account.addPoints(price_house*player.getHouseammount()+price_hotel*player.getHotelammount());
    }


}

```

```java
package deck;

import entity.Player;

public class QueensCard extends Card{

    public QueensCard(String message) {
        super(message);
    }

    @Override
    public void drawCard(Player player) {
        player.setOutofjailcard(true);
    }

}

```

```java
package deck;
import entity.Player;

public class RecieveCard extends Card{
    protected int bonus;

    public RecieveCard(String message, int bonus) {
        super(message);
        this.bonus = bonus;
    }

    public void drawCard(Player player) {
        player.account.addPoints(bonus);
    }


}

```

```java
package deck;
import entity.Player;
import fields.GameBoard;

public class RecieveIfCard extends RecieveCard{
    private int maxvalue;
    private GameBoard board;

    public RecieveIfCard(String message, int bonus, int maxvalue, GameBoard board) {
        super(message, bonus);
        this.maxvalue = maxvalue;
        this.bonus = bonus;
        this.board = board;
    }

    public void drawCard(Player player) {
        int networth;
        int ownableworth = 0;
        int buildingworth = 0;
        for (int i=0; i<40;i++) {
            if (player.equals(board.getField(i).getOwner())) {
                ownableworth += board.getField(i).getPrice();
                buildingworth += (board.getField(i).getNumberofhouses() *
board.getField(i).getHouseprice());
            }
        }
        networth = player.account.getScore() + ownableworth + buildingworth;
        if (networth <= maxvalue)
            player.account.addPoints(bonus);
    }
}

```

```java
package test_JUnit;

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

@RunWith(Suite.class)
@SuiteClasses({ DeckTest.class, HouseTest.class, JailTest.class,
        MoveToJailTest.class, PastStartTest.class, TestPurchase.class })
public class AllTests {

}

```

```java
package test_JUnit;

import static org.junit.Assert.*;
import boundary.GUIController;
import entity.DiceBox;
import entity.Player;
import fields.GameBoard;
import deck.Deck;

import org.junit.Before;
import org.junit.Test;

import control.HouseController;

public class DeckTest {

    @Test
    public void testKort1() {
    //Preconditions
        DiceBox box = new DiceBox();
        GameBoard board = new GameBoard(box);
        Player[] players = new Player[3];
        Deck deck = new Deck(players, board);

        players[0] = new Player("Spiller1");
        players[1] = new Player("Spiller2");
        players[2] = new Player("Spiller3");
        players[0].setPosition(0);
    //Test
        //Move to "Groenningen"
        deck.pickCard(players[0], 0);

    //Postconditions
        //Tjek om spilleren er pï¿½ feltet
        assertEquals(24,players[0].getPosition());

    }

    @Test
    public void testKort2() {
        //Preconditions
            DiceBox box = new DiceBox();
            GameBoard board = new GameBoard(box);
            Player[] players = new Player[3];
            Deck deck = new Deck(players, board);

            players[0] = new Player("Spiller1");
            players[1] = new Player("Spiller2");
            players[2] = new Player("Spiller3");
            players[0].setPosition(7);
        //Test
            //Draws second card, move to nearest shipping company.
            deck.pickCard(players[0], 1);

        //Postconditions
            //Tjek om spilleren er pï¿½ feltet
            assertEquals(15,players[0].getPosition());
        }

    @Test
    public void testKort3(){
    //Preconditions
```

```
63          DiceBox box = new DiceBox();
64          GameBoard board = new GameBoard(box);
65          Player[] players = new Player[3];
66          Deck deck = new Deck(players, board);
67
68          players[0] = new Player("Spiller1");
69          players[1] = new Player("Spiller2");
70          players[2] = new Player("Spiller3");
71
72
73          //We land on chance card at field 8.
74          players[0].setPosition(7);
75
76
77
78      //Test
79          //Draws third card, move to start.
80          deck.pickCard(players[0], 2);
81
82
83
84      //Postconditions
85          //Check if the given field.
86          assertEquals(0,players[0].getPosition());
87      }
88
89      @Test
90      public void testKort4(){
91      //Preconditions
92          DiceBox box = new DiceBox();
93          GameBoard board = new GameBoard(box);
94          Player[] players = new Player[3];
95          Deck deck = new Deck(players, board);
96
97          players[0] = new Player("Spiller1");
98          players[1] = new Player("Spiller2");
99          players[2] = new Player("Spiller3");
100
101
102         //We land on chance card at field 8.
103         players[0].setPosition(7);
104
105
106
107     //Test
108         //Move 3 positions back.
109         deck.pickCard(players[0], 3);
110
111
112
113     //Postconditions
114         //Check if the player is on the given field.
115         assertEquals(4,players[0].getPosition());
116     }
117
118     @Test
119     public void testKort5(){
120     //Preconditions
121         DiceBox box = new DiceBox();
122         GameBoard board = new GameBoard(box);
123         Player[] players = new Player[3];
124         Deck deck = new Deck(players, board);
```

```
125
126         players[0] = new Player("Spiller1");
127         players[1] = new Player("Spiller2");
128         players[2] = new Player("Spiller3");
129
130
131         //We land on chance card at field 18.
132         players[0].setPosition(17);
133
134
135
136     //Test
137         //Move to Frederiksberg Alle
138         deck.pickCard(players[0], 4);
139
140
141
142     //Postconditions
143         //Check if the player is on the given field.
144         assertEquals(11,players[0].getPosition());
145         //Check if player received money when crossing start.
146         assertEquals(34000,players[0].account.getScore());
147     }
148
149     @Test
150     public void testKort6(){
151     //Preconditions
152         DiceBox box = new DiceBox();
153         GameBoard board = new GameBoard(box);
154         Player[] players = new Player[3];
155         Deck deck = new Deck(players, board);
156
157         players[0] = new Player("Spiller1");
158         players[1] = new Player("Spiller2");
159         players[2] = new Player("Spiller3");
160
161
162         //We land on chance card at field 18.
163         players[0].setPosition(36);
164
165
166
167     //Test
168         //Use mols linien, if you cross start receive 4000.
169         deck.pickCard(players[0], 5);
170
171
172
173     //Postconditions
174         //Check if the player is on the given field.
175         assertEquals(15,players[0].getPosition());
176         //Check if player received money when crossing start.
177         assertEquals(34000,players[0].account.getScore());
178     }
179
180     @Test
181     public void testKort8(){
182     //Preconditions
183         DiceBox box = new DiceBox();
184         GameBoard board = new GameBoard(box);
185         Player[] players = new Player[3];
186         Deck deck = new Deck(players, board);
```

```
187
188         players[0] = new Player("Spiller1");
189         players[1] = new Player("Spiller2");
190         players[2] = new Player("Spiller3");
191
192
193         //We land on chance card at field 36.
194         players[0].setPosition(36);
195
196
197
198     //Test
199         //Use mols linien, if you cross start receive 4000.
200         deck.pickCard(players[0], 7);
201
202
203
204     //Postconditions
205         //Check if the player is set to a jailed status.
206         assertEquals(true,players[0].isJailed());
207         //Check if the player is on the jail field.
208         assertEquals(10,players[0].getPosition());
209
210     }
211
212     @Test
213     public void testKort10(){
214     //Preconditions
215         DiceBox box = new DiceBox();
216         GameBoard board = new GameBoard(box);
217         Player[] players = new Player[3];
218         Deck deck = new Deck(players, board);
219
220         players[0] = new Player("Spiller1");
221         players[1] = new Player("Spiller2");
222         players[2] = new Player("Spiller3");
223
224
225         //We land on chance card at field 7.
226         players[0].setPosition(7);
227
228
229
230     //Test
231         //Repair on car, 3000 bill.
232         deck.pickCard(players[0],9);
233
234
235
236     //Postconditions
237         //Check if the player lost 3000.
238         assertEquals(27000,players[0].account.getScore());
239
240     }
241     @Before
242     public void ini(){
243     DiceBox box = new DiceBox();
244     GameBoard board = new GameBoard(box);
245     Player[] players = new Player[3];
246     players[0] = new Player("Spiller1");
247     players[1] = new Player("Spiller2");
248     players[2] = new Player("Spiller3");
```

```
249     GUIController GUIC = new GUIController();
250     HouseController HC = new HouseController(GUIC, board, players);
251     }
252
253     @Test
254     public void testKort11(){
255     //Preconditions
256         DiceBox box = new DiceBox();
257         GameBoard board = new GameBoard(box);
258         Player[] players = new Player[3];
259         players[0] = new Player("Spiller1");
260         players[1] = new Player("Spiller2");
261         players[2] = new Player("Spiller3");
262         GUIController GUIC = new GUIController();
263         HouseController HC = new HouseController(GUIC, board, players);
264         // Preconditions
265         players[0].setPosition(11);
266         board.getField(11).setBuyfield(true);
267         board.getField(players[0].getPosition()).landOnField(players[0]);
268         players[0].setPosition(13);
269         board.getField(13).setBuyfield(true);
270         board.getField(players[0].getPosition()).landOnField(players[0]);
271         players[0].setPosition(14);
272         board.getField(14).setBuyfield(true);
273         board.getField(players[0].getPosition()).landOnField(players[0]);
274         assertEquals(21200, players[0].account.getScore());
275
276         // Test
277         HC.checkOwnedFields(0);
278         HC.buyHouse(0);
279
280         // Postconditions
281         assertEquals(3,players[0].getFieldammount_green());
282         assertEquals(15200, players[0].account.getScore());
283     }
284
285     @Test
286     public void testKort19(){
287         // Giftcard
288
289     //Preconditions
290         DiceBox box = new DiceBox();
291         GameBoard board = new GameBoard(box);
292         Player[] players = new Player[3];
293
294         players[0] = new Player("Spiller1");
295         players[1] = new Player("Spiller2");
296         players[2] = new Player("Spiller3");
297         Deck deck = new Deck(players, board);
298
299         // Preconditions
300         players[0].setPosition(7);
301
302         // Test
303         deck.pickCard(players[0], 19);
304
305         // Postconditions
306         assertEquals(30400, players[0].account.getScore());
307         assertEquals(29800, players[1].account.getScore());
308         assertEquals(29800, players[2].account.getScore());
309     }
310
```

```
311     @Test
312     public void testKort28(){
313     //Preconditions
314         DiceBox box = new DiceBox();
315         GameBoard board = new GameBoard(box);
316         Player[] players = new Player[3];
317         Deck deck = new Deck(players, board);
318 
319         players[0] = new Player("Spiller1");
320         players[1] = new Player("Spiller2");
321         players[2] = new Player("Spiller3");
322 
323         // Preconditions
324         players[0].setPosition(11);
325         board.getField(11).setBuyfield(true);
326         board.getField(players[0].getPosition()).landOnField(players[0]);
327         players[0].setPosition(13);
328         board.getField(13).setBuyfield(true);
329         board.getField(players[0].getPosition()).landOnField(players[0]);
330         players[0].account.setScore(4000);
331 
332 
333 
334         // Test
335         deck.pickCard(players[0], 28);
336 
337         // Postconditions
338         assertEquals(2,players[0].getFieldammount_green());
339         assertEquals(44000, players[0].account.getScore());
340     }
341 }
342 
```

```java
package test_JUnit;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

import boundary.GUIController;
import control.HouseController;
import entity.DiceBox;
import entity.Player;
import fields.GameBoard;

public class HouseTest {

    @Before
    public void ini(){
    DiceBox box = new DiceBox();
    GameBoard board = new GameBoard(box);
    Player[] players = new Player[3];
    players[0] = new Player("Spiller1");
    players[1] = new Player("Spiller2");
    players[2] = new Player("Spiller3");
    GUIController GUIC = new GUIController();
    HouseController HC = new HouseController(GUIC, board, players);
    }

    @Test
    public void testHouse(){
    //Preconditions
        DiceBox box = new DiceBox();
        GameBoard board = new GameBoard(box);
        Player[] players = new Player[3];
        players[0] = new Player("Spiller1");
        players[1] = new Player("Spiller2");
        players[2] = new Player("Spiller3");
        GUIController GUIC = new GUIController();
        HouseController HC = new HouseController(GUIC, board, players);
    // Preconditions
        players[0].setPosition(11);
        board.getField(11).setBuyfield(true);
        board.getField(players[0].getPosition()).landOnField(players[0]);
        players[0].setPosition(13);
        board.getField(13).setBuyfield(true);
        board.getField(players[0].getPosition()).landOnField(players[0]);
        players[0].setPosition(14);
        board.getField(14).setBuyfield(true);
        board.getField(players[0].getPosition()).landOnField(players[0]);
        assertEquals(21200, players[0].account.getScore());
        assertEquals(30000, players[1].account.getScore());
    // Test
        HC.checkOwnedFields(0);
        HC.buyHouse(0);

        players[1].setPosition(11);
        board.getField(players[1].getPosition()).landOnField(players[1]);


    // Postconditions
        assertEquals(3,players[0].getFieldammount_green());
        assertEquals(16200, players[0].account.getScore());
        assertEquals(29000, players[1].account.getScore());
```

```
63
64      }
65  }
66
67
```

```java
package test_JUnit;

import static org.junit.Assert.*;
import entity.DiceBox;
import entity.Player;
import fields.GameBoard;

import org.junit.Test;

import boundary.GUIController;
import control.TurnController;

public class JailTest {

    @Test
    public void testJail1() {
        // Tests the getoutofjailcard
        DiceBox box = new DiceBox();
        GameBoard board = new GameBoard(box);
        Player[] playerlist = new Player[3];
        playerlist[0] = new Player("Spiller1");
        playerlist[1] = new Player("Spiller2");
        playerlist[2] = new Player("Spiller3");
        GUIController GC = new GUIController();
        TurnController TC = new TurnController(GC, board, playerlist,1);

        // Preconditions
        playerlist[0].setOutofjailcard(true);
        playerlist[0].setJailed(true);
        playerlist[0].setPosition(10);

        // Test
        TC.exitCard(0);

        // Postconditions
        assertEquals(false, playerlist[0].isJailed());
    }
    @Test
    public void testJail2() {
        // Tests pay to get out of jail
        DiceBox box = new DiceBox();
        GameBoard board = new GameBoard(box);
        Player[] playerlist = new Player[3];
        playerlist[0] = new Player("Spiller1");
        playerlist[1] = new Player("Spiller2");
        playerlist[2] = new Player("Spiller3");
        GUIController GC = new GUIController();
        TurnController TC = new TurnController(GC, board, playerlist,1);

        // Preconditions
        playerlist[0].setJailed(true);
        playerlist[0].setPosition(10);

        // Test
        TC.exitPay(0);

        // Postconditions
        assertEquals(29000, playerlist[0].account.getScore());
    }
    @Test
    public void testJail3() {
        // Test throw to exit jail
```

```
63          DiceBox box = new DiceBox();
64          GameBoard board = new GameBoard(box);
65          Player[] playerlist = new Player[3];
66          playerlist[0] = new Player("Spiller1");
67          playerlist[1] = new Player("Spiller2");
68          playerlist[2] = new Player("Spiller3");
69          GUIController GC = new GUIController();
70          TurnController TC = new TurnController(GC, board, playerlist,1);
71
72          // Preconditions
73          playerlist[0].setJailed(true);
74          playerlist[0].setPosition(10);
75
76          // First Throw - not pair
77          box.setDice(0, 6);
78          box.setDice(1, 4);
79          TC.exitThrowTest(0, box);
80
81          assertEquals(true, playerlist[0].isJailed());
82
83          // Second Throw - not pair
84          box.setDice(0, 5);
85          box.setDice(1, 1);
86          TC.exitThrowTest(0, box);
87
88          assertEquals(true, playerlist[0].isJailed());
89
90          // Third Throw - pair
91          box.setDice(0, 4);
92          box.setDice(1, 4);
93          TC.exitThrowTest(0, box);
94
95          assertEquals(false, playerlist[0].isJailed());
96          assertEquals(18, playerlist[0].getPosition());
97
98      }
99      @Test
100     public void testJail4() {
101         // Test throw to exit jail, but after 3 turns (9 tries), he is forced to pay
102         DiceBox box = new DiceBox();
103         GameBoard board = new GameBoard(box);
104         Player[] playerlist = new Player[3];
105         playerlist[0] = new Player("Spiller1");
106         playerlist[1] = new Player("Spiller2");
107         playerlist[2] = new Player("Spiller3");
108         GUIController GC = new GUIController();
109         TurnController TC = new TurnController(GC, board, playerlist, 1);
110
111         // Preconditions
112         playerlist[0].setJailed(true);
113         playerlist[0].setPosition(10);
114
115         // Turn 1
116         // First Throw - not pair
117                 box.setDice(0, 6);
118                 box.setDice(1, 4);
119                 TC.exitThrowTest(0, box);
120
121                 assertEquals(true, playerlist[0].isJailed());
122
123         // Second Throw - not pair
124                 box.setDice(0, 5);
```

```
125                 box.setDice(1, 1);
126                 TC.exitThrowTest(0, box);
127
128                 assertEquals(true, playerlist[0].isJailed());
129
130         // Third Throw - not pair
131                 box.setDice(0, 5);
132                 box.setDice(1, 4);
133                 TC.exitThrowTest(0, box);
134
135                 assertEquals(true, playerlist[0].isJailed());
136                 assertEquals(1,playerlist[0].getJailcount());
137
138         // Turn 2
139         // First Throw - not pair
140                 box.setDice(0, 6);
141                 box.setDice(1, 4);
142                 TC.exitThrowTest(0, box);
143
144                 assertEquals(true, playerlist[0].isJailed());
145
146         // Second Throw - not pair
147                 box.setDice(0, 5);
148                 box.setDice(1, 1);
149                 TC.exitThrowTest(0, box);
150
151                 assertEquals(true, playerlist[0].isJailed());
152
153         // Third Throw - not pair
154                 box.setDice(0, 5);
155                 box.setDice(1, 4);
156                 TC.exitThrowTest(0, box);
157
158                 assertEquals(true, playerlist[0].isJailed());
159                 assertEquals(2,playerlist[0].getJailcount());
160
161         // Turn 3
162         // First Throw - not pair
163                 box.setDice(0, 6);
164                 box.setDice(1, 4);
165                 TC.exitThrowTest(0, box);
166
167                 assertEquals(true, playerlist[0].isJailed());
168
169         // Second Throw - not pair
170                 box.setDice(0, 5);
171                 box.setDice(1, 1);
172                 TC.exitThrowTest(0, box);
173
174                 assertEquals(true, playerlist[0].isJailed());
175
176         // Third Throw - not pair
177         // Now the player should be forced to pay
178                 box.setDice(0, 5);
179                 box.setDice(1, 4);
180                 TC.exitThrowTest(0, box);
181
182
183         // Postconditions
184                 assertEquals(false, playerlist[0].isJailed());
185                 assertEquals(29000, playerlist[0].account.getScore());
186                 assertEquals(0,playerlist[0].getJailcount());
```

```
187     }
188
189 }
190
```

MoveToJailTest.java

```java
package test_JUnit;

import static org.junit.Assert.*;

import org.junit.Test;

import entity.Player;
import fields.GameBoard;

public class MoveToJailTest {
    @Test
    public void testFaengsel(){

    //Preconditions
        //Initialise gameboard and player objects.
    GameBoard gameboard = new GameBoard(null);
    Player player1 = new Player("Spiller1");

    //Test
        // Sets the players position to 30.
    player1.setPosition(30);
    gameboard.getField(player1.getPosition()).landOnField(player1);

    //Postconditions
        // Check if it is true that the player is now in jail.
    assertEquals(true, player1.isJailed());
    }
}

```

```java
package test_JUnit;

import static org.junit.Assert.*;

import org.junit.Test;

import entity.Player;


public class PastStartTest {

    @Test
    public void test1() {
    //Preconditions
    Player player1 = new Player("Spiller1");
    //Test
        //Set the player position to field 37.
        player1.setPosition(37);
        //Move player 5 positions forward.
        player1.movePosition(5);
    //Postconditions
        //Check if position 2
        assertEquals(2,player1.getPosition());
    }

    @Test
    public void test2(){
    //Preconditions
        //Initialise the gameboard and player object.
        Player player1 = new Player("Spiller1");
        //Set the player score to 5000.
        player1.account.setScore(5000);

    //Test
        //Set the player position to field 37.
        player1.setPosition(37);
        //Move player 5 positions forward.
        player1.movePosition(5);
    //Postconditions
        //Check if position 2
        assertEquals(2,player1.getPosition());
        //Check if player received 4000 when crossing start.
        assertEquals(9000,player1.account.getScore());
    }


}

```

```java
package test_JUnit;

import static org.junit.Assert.*;
import entity.DiceBox;
import entity.Player;
import fields.GameBoard;
import org.junit.Test;

public class TestPurchase {

    @Test
    public void test() {
    //Preconditions
        DiceBox box = new DiceBox();
        GameBoard board = new GameBoard(box);
        Player[] players = new Player[3];
        players[0] = new Player("Spiller1");
        players[1] = new Player("Spiller2");
        players[2] = new Player("Spiller3");
        players[0].setPosition(0);
    //Test
        //Move to "Groenningen"
        board.getField(24).setBuyfield(true);
        players[0].setPosition(24);
        board.getField(24).landOnField(players[0]);
        assertEquals(25200, players[0].account.getScore());
        players[1].setPosition(24);
        board.getField(24).landOnField(players[1]);

    //Postconditions
        //Tjek om spilleren er pï¿½ feltet
        assertEquals(25600,players[0].account.getScore());
        assertEquals(29600,players[1].account.getScore());

    }
}

```