

```

1 package controller;
2
3 import java.awt.Color;
4
5
6
7 public class GameController {
8     protected Board board; //An instance of the Board class
9     protected Player[] players; //An array of Players
10
11     public GameController(){
12         board = new Board();
13         GUIController.initializeBoard(board);
14     }
15
16     public void resetGame(int playerAmount, int balance){
17         players = new Player[playerAmount];
18         Color color = null;
19
20         for (int i = 0; i < players.length; i++){
21
22             switch (i){
23                 case 0: color = Color.red; break;
24                 case 1: color = Color.green; break;
25                 case 2: color = Color.yellow; break;
26                 case 3: color = Color.blue; break;
27                 case 4: color = Color.white; break;
28                 case 5: color = Color.black; break;
29                 default: System.exit(1);
30             }
31
32             players[i] = new Player(i+1, Messages.getGeneralMessages()[10]+(i+1), new Piece(color), new Account(balance));
33
34             GUIController.addPlayer(players[i].getName(), players[i].getAccount().getBalance(), players[i].getPiece().getColor());
35
36             for (Field j : board.getFields()){
37                 if (j instanceof Ownable){
38                     //remove owner of field
39                     ((Ownable) j).setOwner(null);

```

```

40         GUIController.removeFieldOwner(j.getId());
41
42         //remove houses and hotels
43         if (j instanceof Street){
44             ((Street) j).setHousesOwned(0);
45             GUIController.setHouses(j.getId(),0);
46         }
47     }
48
49 }
50
51 }
52 }
53
54 public void playGame(){
55     boolean winnerFound = false;
56     Player currentPlayer;
57
58     //first player is player 1
59     currentPlayer = players[0];
60
61     while (winnerFound == false){
62
63         TurnController turn = new TurnController(currentPlayer,board);
64         turn.playTurn();
65
66         if (currentPlayer.getAccount().getBalance() < 0){
67             removePlayer(currentPlayer);
68         }
69
70         if (players.length == 1){
71             winnerFound = true;
72             GUIController.showMessage((currentPlayer.getName()) + Messages.getGeneralMessages()[15]);
73         }
74         else{
75             currentPlayer = defineNextPlayer(currentPlayer);
76         }

```

```

77     }
78 }
79
80 protected void removePlayer(Player player){
81     Player[] temp;
82     temp = players;
83
84     players = new Player[temp.length-1];
85
86     int playerCount = 0;
87     for (int i = 0; i<temp.length;i++){
88         if (temp[i] != player){
89             players[playerCount] = temp[i];
90             playerCount++;
91         }
92     }
93
94     GUIController.removeAllCars(player.getName());
95
96     //puts players owned fields back on sale
97     for (Ownable i : player.getAccount().getOwnedFields()){
98         if(i != null){
99             GUIController.removeFieldOwner(i.getId());
100             i.setOwner(null);
101             if (i instanceof Street){
102                 ((Street) i).setHousesOwned(0);
103                 GUIController.setHouses(i.getId(),0);
104             }
105         }
106     }
107
108 }
109
110 protected Player defineNextPlayer(Player currentPlayer){
111     Player nextPlayer;
112
113     //current player threw two equal

```

```
114     if(currentPlayer.getEqualCount() > 0 && currentPlayer.getAccount().getBalance() >= 0){
115         nextPlayer = currentPlayer;
116     }
117     //current player is last player in array
118     else if (currentPlayer == players[players.length-1]){
119         nextPlayer = players[0];
120     }
121     //find next player in array
122     else{
123         int arrayIndex = 0;
124         for (int i=0;i<players.length;i++){
125             if (currentPlayer == players[i]){
126                 arrayIndex=i;
127             }
128         }
129         nextPlayer = players[arrayIndex+1];
130     }
131
132     return nextPlayer;
133 }
134
135 }
136
```

```
1 package controller;
2
3 import java.awt.Color;
4
5
6
7
8
9 public class GUIController {
10
11     public static void initializeBoard(Board board){
12         Field[] fields = board.getFields();
13         desktop_fields.Field[] graphicfields = new desktop_fields.Field[fields.length];
14
15         for(int i = 0; i < fields.length; i++){
16             if (fields[i] instanceof Brewery){
17                 graphicfields[i] = new desktop_fields.Brewery.Builder()
18                     .setTitle(Messages.getFieldNames()[i])
19                     .setDescription(Messages.getFieldNames()[i])
20                     .setSubText(determineSubText(board, i))
21                     .setRent(determineRent(board, i))
22                     .build();
23             }
24             else if (fields[i] instanceof Fleet){
25                 graphicfields[i] = new desktop_fields.Shipping.Builder()
26                     .setTitle(Messages.getFieldNames()[i])
27                     .setDescription(Messages.getFieldNames()[i])
28                     .setSubText(determineSubText(board, i))
29                     .setRent(determineRent(board, i))
30                     .build();
31             }
32             else if (fields[i] instanceof Tax){
33                 graphicfields[i] = new desktop_fields.Tax.Builder()
34                     .setTitle(determineSubText(board, i))
35                     .setDescription(Messages.getFieldNames()[i])
36                     .setSubText(determineSubText(board, i))
37                     .build();
38             }
39             else if (fields[i] instanceof GoToPrison || i == 10){
40                 graphicfields[i] = new desktop_fields.Jail.Builder()
41                     .setTitle(determineSubText(board, i))
```

```

42         .setDescription(Messages.getFieldNames()[i])
43         .setSubText(determineSubText(board, i))
44         .build();
45     }
46     else if (fields[i] instanceof Chance){
47         graphicfields[i] = new desktop_fields.Chance.Builder()
48 //         .setTitle(determineSubText(board, i))
49 //         .setDescription(Messages.getFieldNames()[i])
50 //         .setSubText(determineSubText(board, i))
51         .build();
52     }
53     else if (i == 20){
54         graphicfields[i] = new desktop_fields.Refuge.Builder()
55         .setTitle(Messages.getFieldNames()[i])
56         .setDescription(Messages.getFieldNames()[i])
57         .setSubText(determineSubText(board, i))
58         .build();
59     }
60     else if (i == 0){
61         graphicfields[i] = new desktop_fields.Start.Builder()
62         .setTitle(Messages.getFieldNames()[i])
63         .setDescription(Messages.getFieldNames()[i])
64         .setSubText(determineSubText(board, i))
65         .build();
66     }
67     else{
68         graphicfields[i] = new desktop_fields.Street.Builder()
69         .setBgColor(fields[i].getColor())
70         .setTitle(Messages.getFieldNames()[i])
71         .setDescription(Messages.getFieldNames()[i])
72         .setSubText(determineSubText(board, i))
73         .setRent(determineRent(board, i))
74         .build();
75     }
76
77
78 }

```

```

79
80     GUI.create(graphicfields);
81     GUI.displayChanceCard();
82 }
83
84 public static void addPlayer(String name, int balance, Color pieceColor){
85     Car car = new Car.Builder()
86         .primaryColor(pieceColor)
87         .build();
88     GUI.addPlayer(name, balance, car);
89     GUI.setBalance(name, balance);
90     GUI.setCar(1,name);
91 }
92
93 public static void removeAllCars(String playerName){
94     GUI.removeAllCars(playerName);
95 }
96
97 private static String determineSubText(Board board, int fieldNumber){
98     Field[] fields = board.getFields();
99
100     String text = "";
101     if(fields[fieldNumber] instanceof Ownable){
102         text += Messages.getBoardMessages()[0]; //Price:
103         text += " " + String.valueOf(((Ownable) fields[fieldNumber])).getPrice());
104     }
105     else if(fields[fieldNumber] instanceof Tax){
106         text += Messages.getBoardMessages()[3]; //Pay:
107         text += " " + String.valueOf(((Tax) fields[fieldNumber])).getTaxAmount());
108         if(((Tax) fields[fieldNumber])).getTaxRate() > 0){
109             text += " " + Messages.getBoardMessages()[4];
110             text += " " + String.valueOf(((Tax) fields[fieldNumber])).getTaxRate());
111             text += "% " + Messages.getBoardMessages()[5];
112         }
113     }
114     return text;
115 }

```

# GUIController.java

```
116
117 private static String determineRent(Board board, int fieldNumber){
118     Field[] fields = board.getFields();
119
120     String rent = "";
121     if(fields[fieldNumber] instanceof Ownable){
122         rent += Messages.getGeneralMessages()[22] + String.valueOf((((Ownable) fields[fieldNumber])).getRent());
123     }
124
125     return rent;
126 }
127
128 public static String getUserChoice(String message, String ... options){
129     return GUI.getUserSelection(message, options);
130 }
131
132 public static String getUserButtonPressed(String message, String ... buttons){
133     return GUI.getUserButtonPressed(message, buttons);
134 }
135
136 public static void showMessage(String message){
137     GUI.showMessage(message);
138 }
139
140 public static void setDice(int value1, int value2){
141     GUI.setDice(value1, value2);
142 }
143
144 public static void setCar(int position, String playerName){
145     GUI.setCar(position, playerName);
146 }
147
148 public static void setFieldOwner(String playerName, int fieldNumber){
149     GUI.setOwner(fieldNumber, playerName);
150 }
151
152 public static void removeFieldOwner(int fieldNumber){
```



# GUIController.java

```
153         GUI.removeOwner(fieldNumber);
154     }
155
156     public static void setPlayerBalance(String playerName, int balance){
157         GUI.setBalance(playerName, balance);
158     }
159
160     public static String getUserSelection(String message, String ... options){
161         return GUI.getUserSelection(message, options);
162     }
163
164     public static void setHouses(int fieldNumber, int houses){
165         GUI.setHouses(fieldNumber, houses);
166     }
167
168     public static void setHotel(int fieldNumber){
169         GUI.setHotel(fieldNumber, true);
170     }
171
172 }
173
174
```

```
1 package controller;
2
3 import desktop_resources.GUI;
4
5
6 public class Manno1936 {
7
8     public static void main(String[] args) {
9         GameController game = new GameController(); //Initialize game
10
11         while (GUIController.getUserButtonPressed(Messages.getGeneralMessages()[6] //do you want to create new game?
12             , Messages.getGeneralMessages()[1] //yes
13             , Messages.getGeneralMessages()[2] //no
14             ).equals(Messages.getGeneralMessages()[1]) //user chooses yes
15             ){
16             int playerAmount = Integer.parseInt(GUIController.getUserChoice(Messages.getGeneralMessages()[8], "3", "4", "5", "6"));
17             game.resetGame(playerAmount, 1500); //Reset the game with new amount of players and set start balance
18             game.playGame();
19             GUI.close();
20         }
21         GUI.close();
22         System.exit(0);
23     }
24
25 }
26
```

## TurnController.java

```
1 package controller;
2
3 import java.awt.Color;
4
5 import entity.*;
6
7 public class TurnController {
8     protected DiceCup dice; //An instance of the DiceCup class
9     protected Field currentField;
10    protected Player player;
11    protected Board board;
12
13    protected final int prisonEscapeFine = 50;
14    protected final int payday = 200;
15
16    protected boolean movingToPrison;
17    protected boolean movingPiece;
18
19    public TurnController(Player player, Board board){
20        this.player = player;
21        this.board = board;
22
23        dice = new DiceCup(6,2);
24    }
25
26
27    public void playTurn(){
28        //This used to be in the constructor, but problems arrived when trying to implement
29        //a test class inheriting from this class as it requires the super class constructor to be used,
30        //which in turn prevents us from doing any of our own determined inputs
31
32        do{
33
34            boolean buyHouseChoice = false;
35
36            //we go through owned buildable streets
37            for (Street ownedStreet : player.getAccount().getBuildableStreets()){
```

```

38
39     Color fieldColor = ownedStreet.getColor();
40
41     //we find number of streets in group
42     int groupAmount = 0;
43     for (Field boardField : board.getFields()){
44         if (boardField.getColor() == fieldColor && boardField instanceof Street){
45             groupAmount++;
46         }
47     }
48
49     //we find streets in group owned by player
50     int ownedInGroup = 0;
51     for(Ownable ownedField : player.getAccount().getOwnedFields()){
52         if (ownedField.getColor() == fieldColor && ownedField instanceof Street){
53             ownedInGroup++;
54         }
55     }
56
57     //we find out if player has all streets in group
58     if (ownedInGroup == groupAmount){ //player owns all in group
59         buyHouseChoice = true;
60         break;
61     }
62 }
63
64 //Player can buy houses
65 if (buyHouseChoice == true){
66
67     //Do you want to throw dice or buy houses/hotels
68     player.setChoice(determineUserInput(new String[]{Messages.getGeneralMessages()[11] + player.getName() +
Messages.getGeneralMessages()[12],
69         Messages.getGeneralMessages()[7],
70         Messages.getGeneralMessages()[21]}}));
71
72     //Player wants to buy house or hotel
73     if (player.getChoice().equals(Messages.getGeneralMessages()[21])){

```

```

74         buyHouseHotel();
75     }
76 }
77 //Player can only throw dice
78 else{
79     player.setChoice(determineUserInput(new String[]{Messages.getGeneralMessages()[11] + player.getName() +
Messages.getGeneralMessages()[12],
80         Messages.getGeneralMessages()[7]}));
81 }
82 }
83 while(!player.getChoice().equals(Messages.getGeneralMessages()[7])); //player does not want to throw dice
84
85 movingPiece = false; // A boolean to define whether or not a piece shall move. I saw it necessary to simplify the code.
86
87 // Player is not in prison
88 if (player.getPrisonCount() == 0) {
89     throwDice();
90     movingPiece = true;
91 }
92 // If player is in Prison
93 else if (player.getPrisonCount() > 0) {
94     prisonEscape();
95 }
96
97 if (movingPiece) {
98     movePiece();
99     landOnField();
100     if((currentField instanceof Chance) && (Chance.isMoveCard() == true)){
101         GUIController.removeAllCars(player.getName());
102         currentField = board.getFields()[player.getPiece().getPosition()-1];
103         GUIController.setCar(player.getPiece().getPosition(), player.getName());
104         landOnField();
105     }
106 }
107 }
108
109 protected void throwDice(){

```

```

110     dice.throwDice();
111     player.setLastThrow(dice);
112     GUIController.setDice(dice.getDice()[0].getValue(), dice.getDice()[1].getValue());
113
114     //Are dice equal?
115     if(dice.isEqual() == true){
116         //Player has thrown equals less than 3 times in a row
117         if (player.getEqualCount() != 2){
118             player.setEqualCount(player.getEqualCount()+1);
119             movingToPrison = false;
120         }
121         //Player has thrown equals 3 times in a row
122         else{
123             determineUserInput(new String[]{Messages.getGeneralMessages()[29], Messages.getGeneralMessages()[29]});
124             movingToPrison = true;
125             player.setEqualCount(0);
126         }
127     }
128     else{
129         player.setEqualCount(0);
130         movingToPrison = false;
131     }
132
133 }
134
135 protected void buyHouseHotel(){
136
137     Street[] ownedStreets = player.getAccount().getBuildableStreets();
138
139     //we go through owned buildable streets
140     for (Street ownedStreet : player.getAccount().getBuildableStreets()){
141
142         Color fieldColor = ownedStreet.getColor();
143
144         //we find number of streets in group and min and max houses
145         int groupAmount = 0;
146         int maxHouses = 0;

```

```

147     int minHouses = 5;
148     for (Field boardField : board.getFields()){
149         if (boardField.getColor() == fieldColor && boardField instanceof Street){
150             groupAmount++;
151
152             //we find current min amount of houses in group
153             if(((Street) boardField).getHousesOwned() < minHouses){
154                 minHouses = ((Street) boardField).getHousesOwned();
155             }
156
157             //we find current max amount of houses in group
158             if(((Street) boardField).getHousesOwned() > maxHouses){
159                 maxHouses = ((Street) boardField).getHousesOwned();
160             }
161         }
162     }
163
164     //we find streets in group owned by player
165     int ownedInGroup = 0;
166     for(Ownable ownedField : player.getAccount().getOwnedFields()){
167         if (ownedField.getColor() == fieldColor && ownedField instanceof Street){
168             ownedInGroup++;
169         }
170     }
171
172     //we find out if player has all streets in group
173     if (ownedInGroup == groupAmount //player owns all in group
174         && (ownedStreet.getHousesOwned() < maxHouses //field has less houses than other field in group
175         || (ownedStreet.getHousesOwned() == maxHouses && minHouses == maxHouses))) { //all fields have same number of
houses
176         //everything is good
177     }
178     //we remove street from ownedStreets
179     else{
180         Street[] oldArray = ownedStreets;
181         ownedStreets = new Street[oldArray.length-1];
182

```

```

183         int counter = 0;
184         for (Street i : oldArray){
185             if (i != ownedStreet){
186                 ownedStreets[counter] = i;
187                 counter++;
188             }
189         }
190     }
191 }
192
193 //we find names of the streets
194 String[] fieldNames = new String[ownedStreets.length];
195
196 for (int i = 0; i < fieldNames.length; i++){
197     fieldNames[i] = Messages.getFieldNames()[ownedStreets[i].getId()-1];
198 }
199
200 //For which street do you want a house/hotel?
201 String userChoice = GUIController.getUserSelection(Messages.getGeneralMessages()[30], fieldNames);
202
203 Street chosenField = null;
204 for (Field i : board.getFields()){
205     if (Messages.getFieldNames()[i.getId()-1].equals(userChoice)){
206         chosenField = (Street) i;
207     }
208 }
209
210 //Are you sure?
211 String areYouSure = determineUserInput(new String[]{Messages.getGeneralMessages()[23] + chosenField.getHousePrice() +
Messages.getGeneralMessages()[24]
212     , Messages.getGeneralMessages()[1], Messages.getGeneralMessages()[2]});
213
214 if (areYouSure.equals(Messages.getGeneralMessages()[1])){
215     chosenField.setHousesOwned(chosenField.getHousesOwned()+1);
216
217     if (chosenField.getHousesOwned() == 5){
218         GUIController.setHotel(chosenField.getId());

```



```

219     }
220     else{
221         GUIController.setHouses(chosenField.getId(),chosenField.getHousesOwned());
222     }
223
224     player.getAccount().setBalance(player.getAccount().getBalance()-chosenField.getHousePrice());
225     GUIController.setPlayerBalance(player.getName(), player.getAccount().getBalance());
226 }
227
228 }
229
230 protected void movePiece(){
231     int oldPosition = player.getPiece().getPosition(); // Save the old player position
232     int position;
233
234     /*
235     * If there has already been placed a car, we remove it before placing a new one
236     */
237     GUIController.removeAllCars(player.getName());
238
239     // Make sure the player did not throw 3 equals in a row.
240     if (movingToPrison == false) {
241         /*
242         * Position is equal to the current position + the sum of the dice throw
243         * We use modulus to calculate whether the player would end up outside the board
244         */
245         position = (player.getPiece().getPosition() + dice.getSum()) % board.getFields().length;
246         //Since we use mod 40, we need to have a special case for field 40, or else we get position == 0
247         if (position == 0){
248             position = board.getFields().length;
249         }
250
251         //We set the car and piece position to the new values
252         player.getPiece().setPosition(position);
253         GUIController.setCar(player.getPiece().getPosition(),player.getName());
254         currentField = board.getFields()[position-1];
255     }

```

# TurnController.java

```

256         // Money when passing or landing on start
257         if (oldPosition > position) {
258             if (position == 1) {
259                 determineUserInput(new String[]{
260                     Messages.getGeneralMessages()[26] + Messages.getFieldNames()[0] + // You landed on Start
261                     Messages.getGeneralMessages()[28] + payday}); // and receives the payday
                amount (200)
262             } else {
263                 determineUserInput(new String[]{Messages.getGeneralMessages()[27] + Messages.getFieldNames()[0] + // You
                passed start
264                     Messages.getGeneralMessages()[28] + payday}); // and
                receives the payday amount (200)
265             }
266             player.getAccount().setBalance(player.getAccount().getBalance() + payday);
267             GUIController.setPlayerBalance(player.getName(), player.getAccount().getBalance());
268         }
269     }
270 }
271 else{
272     currentField = board.getFields()[10];
273     moveToPrison(); // Moves the player and its piece straight to prison
274 }
275 }
276 }
277
278 protected void landOnField(){
279     // You landed on
280     if (player.getPiece().getPosition()-1 != 0) { // No need to tell that you landed on start, when the MovePiece says it
281         determineUserInput(new String[]{Messages.getGeneralMessages()[26] + Messages.getFieldNames()
282         [(player.getPiece().getPosition())-1]});
283     }
284     if (player.getPiece().getPosition()-1 == 30) { // goToPrison field.
285         GUIController.removeAllCars(player.getName());
286         moveToPrison();
287         determineUserInput(new String[]{Messages.getGeneralMessages()[29]});
288     }

```

```

289
290     int playerBalance = player.getAccount().getBalance();
291
292     //Ownable
293     if (currentField instanceof Ownable) {
294         Player owner = ((Ownable) currentField).getOwner();
295         int price = ((Ownable) currentField).getPrice();
296
297         // Do you wish to buy it?
298         if (owner == null && playerBalance >= price) {
299             String playerChoice = determineUserInput(new String[]{
300                 Messages.getGeneralMessages()[0] + ((Ownable) currentField).getPrice() + "?", //Do you want to buy field?
301                 Messages.getGeneralMessages()[1],    // Yes
302                 Messages.getGeneralMessages()[2]      // No
303             });
304
305             player.setChoice(playerChoice);
306
307             if (playerChoice.equals(Messages.getGeneralMessages()[1])) { // User chooses yes
308                 GUIController.setFieldOwner(player.getName(), player.getPiece().getPosition());
309             }
310         }
311         // You don't have enough money to buy field
312         else if (owner == null && playerBalance < price){
313             determineUserInput(new String[]{player.getName() + ": " + Messages.getGeneralMessages()[25]});
314         }
315         // You own the field
316         else if (owner == player){
317             determineUserInput(new String[]{player.getName() + ": " + Messages.getGeneralMessages()[20]});
318         }
319         // You have to pay rent
320         else if (owner != player){
321             int rent = 0;
322             if (currentField instanceof Brewery){
323                 //when brewery we should multiply dice sum with 4 or 10, depending on the amount of owned brewery from the
same owner.
324                 rent = ((Ownable) currentField).getRent()*player.getLastThrow().getSum();

```

```

325         }
326         else{
327             rent = ((Ownable) currentField).getRent();
328         }
329
330         determineUserInput(new String[]{player.getName() + ": " + Messages.getGeneralMessages()[9] + rent +
Messages.getGeneralMessages()[16]});
331     }
332 }
333
334 //Tax
335 else if( currentField instanceof Tax) {
336     if ( ((Tax) currentField).getTaxRate() > 0) {
337
338         String playerChoice = determineUserInput(new String[]{
339             Messages.getGeneralMessages()[3],
340             Messages.getGeneralMessages()[4] + ((Tax) currentField).getTaxRate() + Messages.getGeneralMessages()[5],    //
Percent taxes of all assets
341             Messages.getGeneralMessages()[4] + ((Tax) currentField).getTaxAmount()});    // Fixed amount of tax
342
343         player.setChoice(playerChoice);
344     }
345 }
346 //Chance
347 else if( currentField instanceof Chance){
348     determineUserInput(new String[]{Messages.getChanceMessages()[Chance.getCardID()]});
349 }
350
351 currentField.landOnField(player);
352 GUIController.setPlayerBalance(player.getName(),player.getAccount().getBalance());
353 if (currentField instanceof Ownable){
354     if (((Ownable) currentField).getOwner() != null){
355         GUIController.setPlayerBalance(((Ownable) currentField).getOwner().getName(),((Ownable)
currentField).getOwner().getAccount().getBalance());
356     }
357 }
358 }

```

```
359
360
361 protected String determineUserInput(String[] input){
362     String text;
363     switch(input.length) {
364         case 1:
365             GUIController.showMessage(input[0]);
366             text = Messages.getGeneralMessages()[13]; //OK
367             break;
368         case 2:
369             text = GUIController.getUserButtonPressed(input[0], input[1]);
370             break;
371         case 3:
372             text = GUIController.getUserButtonPressed(input[0], input[1], input[2]);
373             break;
374         case 4:
375             text = GUIController.getUserButtonPressed(input[0], input[1], input[2], input[3]);
376             break;
377         default:
378             text = "";
379             break;
380     }
381     return text;
382 }
383
384 protected void moveToPrison() {
385     GUIController.removeAllCars(player.getName());
386     player.getPiece().setPosition(11);
387     player.setPrisonCount(3);
388     GUIController.setCar(player.getPiece().getPosition(), player.getName());
389     player.setEqualCount(0);
390     movingToPrison = false;
391 }
392
393 protected void prisonEscape() {
394     // Player is in prison and has several attempts to get out.
395     if (player.getPrisonCount() > 1) {
```

```

396     String playerChoice = determineUserInput(new String[]{
397         Messages.getGeneralMessages()[32],      //"De er i fængsel. For at komme ud kan de betale 50 kr. eller slå to
ens"
398         Messages.getGeneralMessages()[31],      //"Betalt 50 kr."
399         Messages.getGeneralMessages()[7]});      //"Slå med terningerne"
400
401     player.setChoice(playerChoice);
402
403     //Player wants to throw dice
404     if(playerChoice.equals(Messages.getGeneralMessages()[7])){
405         throwDice();
406
407         if (dice.isEqual() == true) {
408             player.setEqualCount(1);
409             player.setPrisonCount(0);
410             movingPiece = true;
411         }else {
412             player.setEqualCount(0);
413             player.setPrisonCount(player.getPrisonCount()-1);
414         }
415     }
416     //Player wants to pay fine
417     else if(playerChoice.equals(Messages.getGeneralMessages()[31])){
418         player.setEqualCount(0);
419         player.setPrisonCount(0);
420         player.getAccount().setBalance(player.getAccount().getBalance()-prisonEscapeFine); // pay the fine for getting out
of prison
421         GUIController.setPlayerBalance(player.getName(), player.getAccount().getBalance());
422         throwDice();
423         movingPiece = true;
424     }
425 }
426 // Player is using his last attempt to get out
427 else if(player.getPrisonCount() == 1) {
428     throwDice();
429     if (dice.isEqual() == true) {
430         player.setEqualCount(1);

```

```
431         player.setPrisonCount(0);
432     }
433     else {
434         player.setEqualCount(0);
435         player.setPrisonCount(0);
436         player.getAccount().setBalance(player.getAccount().getBalance()-prisonEscapeFine); // pay the fine for getting out
of prison
437         GUIController.setPlayerBalance(player.getName(), player.getAccount().getBalance());
438     }
439     movingPiece = true;
440 }
441 }
442
443 protected void setDice(DiceCup dice){
444     this.dice = dice;
445 }
446 }
447
```

## Account.java

```
1 package entity;
2
3 public class Account {
4     //not implemented private int jailFreeCounter;
5     private int balance;
6     private Ownable[] ownedFields;
7
8     public Account(int startBalance){
9         //jailFreeCounter = 0;
10        this.balance = startBalance;
11
12        ownedFields = new Ownable[0];
13    }
14
15    public int calculateAssets(){
16        int assets = balance;
17
18        for (Ownable i : ownedFields){
19            if (i != null){
20                assets += i.getPrice();
21            }
22        }
23        return assets;
24    }
25
26    public void setBalance(int balance){
27        this.balance = balance;
28    }
29
30    public int getBalance(){
31        return balance;
32    }
33
34 // public void setJailFreeCounter(int jailFreeCounter){
35 //     this.jailFreeCounter = jailFreeCounter;
36 // }
37
38 // public int getJailFreeCounter(){
39 //     return jailFreeCounter;
40 // }
```



Account.java

```
41
42 public Ownable[] getOwnedFields(){
43     return ownedFields;
44 }
45
46 public void setOwnedField(Ownable field){
47     //add new Ownable to Ownable array
48
49     Ownable[] temp = ownedFields;
50     ownedFields = new Ownable[temp.length+1];
51
52     //insert previous owned fields into new array
53     for (int i = 0; i<temp.length;i++){
54         ownedFields[i] = temp[i];
55     }
56
57     //insert new owned field into new array
58     ownedFields[temp.length] = field;
59
60 }
61
62 public Street[] getBuildableStreets(){
63
64     //find number of streets
65     int streetCounter = 0;
66     for (Ownable i : ownedFields){
67         if (i instanceof Street
68             && ((Street) i).getHousesOwned() < 5
69             && balance >= ((Street) i).getHousePrice()){
70             streetCounter++;
71         }
72     }
73
74     //create new Street array
75     Street[] ownedStreets = new Street[streetCounter];
76
77     streetCounter = 0;
78     for (int i = 0; i < ownedFields.length; i++){
79         if (ownedFields[i] instanceof Street
80             && ((Street) ownedFields[i]).getHousesOwned() < 5
```

Account.java

```
81         && balance >= ((Street) ownedFields[i]).getHousePrice()){
82         ownedStreets[streetCounter] = (Street) ownedFields[i];
83         streetCounter++;
84     }
85 }
86
87     return ownedStreets;
88 }
89 }
90
```

## Board.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public class Board {
6     private Field[] fields;
7
8     public Board(){
9
10         fields = new Field[40];
11
12 /**/         fields[0] = new Refuge(1, Color.white); //Start
13         fields[1] = new Street(2, Color.cyan, 60, new int[]{2,10,30,90,160,250},50); //Rødovrevej
14 /**/         fields[2] = new Chance(3, Color.white); //Chance
15         fields[3] = new Street(4, Color.cyan, 60, new int[]{4,20,60,180,320,540},50); //Hvidovre
16         fields[4] = new Tax(5, Color.white, 10, 200); //Betal Indkomstskat
17         fields[5] = new Fleet(6, Color.white, 200); //Øresund A/S
18         fields[6] = new Street(7, Color.pink, 100, new int[]{6,30,90,270,400,550},50); //Roskildevej
19 /**/         fields[7] = new Chance(8, Color.white); //Chance
20         fields[8] = new Street(9, Color.pink, 100, new int[]{6,30,90,270,400,550},50); //Valby Langgade
21         fields[9] = new Street(10, Color.pink, 120, new int[]{8,40,100,300,450,600},50); //Allégade
22 /*MISSING*/         fields[10] = new Refuge(11, Color.white); //Fængsel
23         fields[11] = new Street(12, Color.green, 140, new int[]{10,50,150,450,625,750},100); //Frederiksberg Allé
24         fields[12] = new Brewery(13, Color.white, 150); //Tuborg
25         fields[13] = new Street(14, Color.green, 140, new int[]{10,50,150,450,625,750},100); //Bülowsvej
26         fields[14] = new Street(15, Color.green, 160, new int[]{12,60,180,500,700,900},100); //Gl. Kongevej
27         fields[15] = new Fleet(16, Color.white, 200); //D.F.D.S
28         fields[16] = new Street(17, Color.gray, 180, new int[]{14,70,200,550,750,950},100); //Bernstorffsvej
29 /**/         fields[17] = new Chance(18, Color.white); //Chance
30         fields[18] = new Street(19, Color.gray, 180, new int[]{14,70,200,550,750,950},100); //Hellerupvej
31         fields[19] = new Street(20, Color.gray, 200, new int[]{16,80,220,600,800,1000},100); //Strandvej
32         fields[20] = new Refuge(21, Color.white); //Helle
33         fields[21] = new Street(22, Color.red, 220, new int[]{18,90,250,700,875,1050},150); //Trianglen
34 /**/         fields[22] = new Chance(23, Color.white); //Chance
35         fields[23] = new Street(24, Color.red, 220, new int[]{18,90,250,700,875,1050},150); //Østerbrogade
36         fields[24] = new Street(25, Color.red, 240, new int[]{20,100,300,750,925,1100},150); //Grønningen
37         fields[25] = new Fleet(26, Color.white, 200); //Ø.K.
38         fields[26] = new Street(27, Color.magenta, 260, new int[]{22,110,330,800,975,1150},150); //Bredgade
39         fields[27] = new Street(28, Color.magenta, 260, new int[]{22,110,330,800,975,1150},150); //Kongens Nytorv
40         fields[28] = new Brewery(29, Color.white, 150); //Carlsberg
```

## Board.java

```
41     fields[29] = new Street(30, Color.magenta, 280, new int[]{22,120,360,850,1025,1200},150); //Østergade
42     fields[30] = new GoToPrison(31, Color.white); //Politi (GoToPrison)
43     fields[31] = new Street(32, Color.yellow, 300, new int[]{26,130,390,900,1100,1275},200); //Amagertorv
44     fields[32] = new Street(33, Color.yellow, 300, new int[]{26,130,390,900,1100,1275},200); //Vimmelskaftet
45 /**/     fields[33] = new Chance(34, Color.white); //Chance
46     fields[34] = new Street(35, Color.yellow, 320, new int[]{28,150,450,1000,1200,1400},200); //Nygade
47     fields[35] = new Fleet(36, Color.white, 200); //D/S Bornholm
48 /**/     fields[36] = new Chance(37, Color.white); //Chance
49     fields[37] = new Street(38, Color.orange, 350, new int[]{35,175,500,1100,1300,1500},200); //Frederiksberggade
50     fields[38] = new Tax(39, Color.white, 0, 100); //Ekstraordinær statsskat
51     fields[39] = new Street(40, Color.orange, 400, new int[]{50,200,600,1400,1700,2000},200); //Rådhuspladsen
52
53 }
54
55 public Field[] getFields(){
56     return fields;
57 }
58 }
59
```

## Brewery.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public class Brewery extends Ownable {
6
7     private final int FACTOR_1 = 4;
8     private final int FACTOR_2 = 10;
9
10    public Brewery(int id, Color color, int price) {
11        super(id, color, price);
12    }
13
14    @Override
15    public int getRent() {
16        int ownedBreweries = 0;
17        int rent = 0;
18
19        if (this.getOwner() != null){
20            for (Ownable i : this.getOwner().getAccount().getOwnedFields()){
21
22                if (i instanceof Brewery){
23                    ownedBreweries++;
24                }
25            }
26
27        }
28
29        switch (ownedBreweries){
30            case 1: rent = FACTOR_1; break;
31            case 2: rent = FACTOR_2; break;
32            default: rent = 0;
33        }
34
35        return rent;
36    }
37 }
38
39 }
40
```

## Chance.java

```

1 package entity;
2
3 import java.awt.Color;
4
5 public class Chance extends Field {
6
7     public Chance(int id, Color color) {
8         super(id, color);
9         shuffleDeck();
10    }
11    //All cardtypes are identified by an integer
12    private static int[] chanceCards = {
13
14 //not implemented// 0: "Ryk brikken frem til det nærmeste dampskibsselskab og betal ejeren to gange den leje, han ellers er berettiget
    til. Hvis selskabet ikke ejes af nogen, kan De købe det af banken.", // Two of these
15         1, // "Tag med Øresundsåden - Flyt brikken frem, og hvis De passerer >>Start<<, indkassér kr. 200,00.",
16         2, // "Ryk frem til Frederiksberg Allé. Hvis De passerer >>Start<<, indkassér kr. 200,00.",
17         3, // "Ryk frem til Grønningen. Hvis De passerer >>Start<<, indkassér da kr. 200,00.",
18         4, // "Tag ind på Rådhuspladsen.",
19         5, // "Ryk frem til >>Start<<.",
20         6, 6, // "Ryk tre felter tilbage.", // Two of these
21         // Property charges
22 //not implemented// 7: "Ejendomsskatterne er steget, ekstraudgifterne er: kr. 50,00 pr. hus, kr. 125,00 pr. hotel.",
23 //not implemented// 8: "Kul- og kokspriserne er steget, og De skal betale: kr. 25,00 pr. hus, kr. 125,00 pr. hotel.",
24         // Expenses
25         9, // "De har kørt frem for >>Fuld Stop<<. Betal kr. 100,00 i bøde.",
26         10, // "De har anskaffet et nyt dæk til Deres vogn. Indbetal kr. 100,00.",
27         11, // "Betal kr. 75,00 for modtagne 2 kasser øl.",
28         12, // "De har måttet vedtage en parkeringsbøde. Betal kr. 20,00 til banken.",
29         13, // "Betal for vognvask og smøring kr. 10,00.",
30         14, // "De har været en tur i udlandet og haft for mange cigaretter med hjem. - Betal told kr. 20,00.",
31         // Prison
32         15, 15, // "Gå i fængsel. Ryk direkte til fængslet. Selv om De passerer >>Start<<, indkasserer De ikke kr. 200,00.", // Two
    of these
33         // Prison mercy
34 //not implemented// 16: "I anledning af Kongens fødselsdag benådes De herved for fængsel. Dette kort kan opbevares, indtil De får brug
    for det, eller De kan sælge det.", // Two of these
35         // For the needy
36         17, // "De modtager >>Matador-legatet for værdig trængende<<, stort kr. 2000,00. Ved værdig trængende forstås, at Deres
    formue, d.v.s. Deres kontante penge + skøder + bygninger, ikke overstiger kr. 750,00.",

```

## Chance.java

```
37         // Bonuses
38         18, // "Deres præmieobligation er kommet ud. De modtager kr. 100,00 af banken.",
39         19, // "Værdien af egen avl fra nyttehaven udgør kr. 200,00, som De modtager af banken.",
40         20, // "Efter auktionen på Assistenthuset, hvor De havde pantsat Deres tøj, modtager De ekstra kr. 108,00.",
41         21, // "De har rettidigt afleveret Deres abonnementskort. Depositum kr. 1,00 udbetales Dem af banken.",
42         22, // "Modtag udbytte af Deres aktier: kr. 50,00.",
43         23, // "Manufakturvarerne er blevet billigere og bedre, herved sparer De kr. 50,00, som De modtager af banken.",
44         24, // "Kommunen har eftergivet et kvartals skat, hæv i banken til en glad aften kr. 150,00.",
45         25, // "De har solgt Deres gamle klude. Modtag kr. 20,00.",
46         26, // "Grundet på dyrtiden har De fået gageforhøjelse. Modtag kr 25,00.",
47         // Money collector
48 //not implemented// 27: "De har lagt penge ud til sammenskudsgilde. Mærkværdigvis betaler alle straks. Modtag fra hver medspiller kr.
49 //25,00."
```

```
49     };
50
51     @Override
52     public void landOnField(Player player) {
53         switch(chanceCards[0]){
54 //not implemented     case 0:
55 //         break;
56         case 1: passedStart(player, 6);
57                 player.getPiece().setPosition(6);
58                 break;
59         case 2: passedStart(player, 12);
60                 player.getPiece().setPosition(12);
61                 break;
62         case 3: passedStart(player, 25);
63                 player.getPiece().setPosition(25);
64                 break;
65         case 4: player.getPiece().setPosition(40);
66                 break;
67         case 5: passedStart(player, 1);
68                 player.getPiece().setPosition(1);
69                 break;
70         case 6: if(player.getPiece().getPosition() == 3){ //if the player draws this card on the first chanceCard field
71                 player.getPiece().setPosition(40); //move to position 40
72             }
73             else{
74                 player.getPiece().setPosition(player.getPiece().getPosition()-3); //ryk 3 felter tilbage
75             }
76         }
```

Chance.java

```
76         break;
77 //not implemented    case 7:
78 //                break;
79 //not implemented    case 8:
80 //                break;
81     case 9: player.getAccount().setBalance(player.getAccount().getBalance()-100);
82         break;
83     case 10: player.getAccount().setBalance(player.getAccount().getBalance()-100);
84         break;
85     case 11: player.getAccount().setBalance(player.getAccount().getBalance()-75);
86         break;
87     case 12: player.getAccount().setBalance(player.getAccount().getBalance()-20);
88         break;
89     case 13: player.getAccount().setBalance(player.getAccount().getBalance()-10);
90         break;
91     case 14: player.getAccount().setBalance(player.getAccount().getBalance()-20);
92         break;
93     case 15: player.getPiece().setPosition(11); //move player to prison
94             player.setPrisonCount(3);
95         break;
96 //not implemented    case 16:
97 //                break;
98     case 17: if(player.getAccount().calculateAssets() <= 750){
99             player.getAccount().setBalance(player.getAccount().getBalance()+2000);
100         }
101         break;
102     case 18: player.getAccount().setBalance(player.getAccount().getBalance()+100);
103         break;
104     case 19: player.getAccount().setBalance(player.getAccount().getBalance()+200);
105         break;
106     case 20: player.getAccount().setBalance(player.getAccount().getBalance()+108);
107         break;
108     case 21: player.getAccount().setBalance(player.getAccount().getBalance()+1);
109         break;
110     case 22: player.getAccount().setBalance(player.getAccount().getBalance()+50);
111         break;
112     case 23: player.getAccount().setBalance(player.getAccount().getBalance()+50);
113         break;
114     case 24: player.getAccount().setBalance(player.getAccount().getBalance()+150);
115         break;
```



# Chance.java

```
116         case 25:player.getAccount().setBalance(player.getAccount().getBalance()+20);
117             break;
118         case 26:player.getAccount().setBalance(player.getAccount().getBalance()+25);
119             break;
120 //not implemented      case 27:
121 //                    break;
122     }
123 //    if(chanceCards[0] == 16){ //When a jailFree card is drawn, it should be removed from the deck
124 //        removeCard();
125 //    }
126 //    else{
127 //        moveCardToBottom();
128 //    }
129
130 }
131
132 public void addCard(int cardID){ //adds a card to the bottom of the deck
133     int[] tempDeck = new int[chanceCards.length+1];
134     for(int i = 0; i < tempDeck.length; i++){
135         if(i < tempDeck.length-1){
136             tempDeck[i] = chanceCards[i];
137         }
138         else{
139             tempDeck[i] = cardID;
140         }
141     }
142     chanceCards = tempDeck;
143 }
144
145 private void removeCard(){ //removes the top card of the deck
146     int[] tempDeck = new int[chanceCards.length-1];
147     for(int i = 1; i < chanceCards.length; i++){
148         tempDeck[i-1] = chanceCards[i];
149     }
150     chanceCards = tempDeck;
151 }
152
153 private void moveCardToBottom(){ //moves the top card of the deck to the bottom
154     addCard(chanceCards[0]);
155     removeCard();
```

# Chance.java

```

156     }
157
158     private void shuffleDeck(){
159         for(int i = 0; i < chanceCards.length; i++){
160             int ranNum = ( (int) (Math.random()*(chanceCards.length)));
161             int currentCard = chanceCards[i];
162             chanceCards[i] = chanceCards[ranNum];
163             chanceCards[ranNum] = currentCard;
164         }
165     }
166     private void passedStart(Player player, int newPosition){
167         int oldPosition = player.getPiece().getPosition();
168         if(oldPosition > newPosition){                                     //did the player pass start?
169             player.getAccount().setBalance(player.getAccount().getBalance()+200); //The player receives 200
170         }
171     }
172     public static int getCardID(){
173         return chanceCards[0];
174     }
175
176     public static boolean isMoveCard(){
177         boolean isMoveCard = false;
178         switch(chanceCards[chanceCards.length-1]){
179             case 1: isMoveCard = true;
180                     break;
181             case 2: isMoveCard = true;
182                     break;
183             case 3: isMoveCard = true;
184                     break;
185             case 4: isMoveCard = true;
186                     break;
187             case 5: isMoveCard = true;
188                     break;
189             case 6: isMoveCard = true;
190                     break;
191             case 15: isMoveCard = true;
192                     break;
193             default: isMoveCard = false;
194         }
195         return isMoveCard;

```

Chance.java

```
196     }  
197  
198     public void setDeck(int[] data){  
199         chanceCards = data;  
200     }  
201  
202 }  
203
```

## Dice.java

```
1 package entity;
2
3 public class Dice {
4     protected int value;
5     protected int sides;
6
7     //Constructor to set amount of dice-sides and the amount of dices
8     //furthermore uses the setAllValuesRandom method which simulates a roll
9     public Dice(int diceSides){
10         this.sides = diceSides;
11         this.setRandom();
12     }
13
14     public int getValue(){
15         return value;
16     }
17
18     //Simulates a roll of the chosen dice(s)
19     public void setRandom(){
20         value = ( (int) (Math.random()*sides)+1);
21     }
22
23     public void setValue(int value){
24         this.value = value;
25     }
26 }
27
```

## DiceCup.java

```
1 package entity;
2
3 public class DiceCup {
4     protected Dice[] dice;
5
6     //Constructor to set amount of dice-sides and the amount of dices
7     //furthermore uses the setAllValuesRandom method which simulates a roll
8     public DiceCup(int diceSides, int diceAmount){
9         dice = new Dice[diceAmount];
10
11         for (int i = 0; i<dice.length; i++){
12             dice[i] = new Dice(diceSides);
13         }
14
15         throwDice();
16     }
17
18     public Dice[] getDice(){
19         return dice;
20     }
21
22     public int getSum(){
23         int sum = 0;
24         for (Dice i : dice){
25             sum += i.getValue();
26         }
27         return sum;
28     }
29
30     public void throwDice(){
31         for (Dice i : dice){
32             i.setRandom();
33         }
34     }
35
36     public boolean isEqual(){
37         boolean isEqual = true;
38         int value = dice[0].getValue();
39
40         //Checks if the first value of dice is equal to the rest
```

## DiceCup.java

```
41     for (Dice i : dice){
42         if (value != i.getValue()){
43             isEqual = false;
44         }
45     }
46
47     return isEqual;
48 }
49
50 }
51
```

## Field.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public abstract class Field {
6     protected int id;
7     protected Color color;
8
9     public Field(int id, Color color){
10         this.id = id;
11         this.color = color;
12     }
13
14     public Color getColor(){
15         return color;
16     }
17
18     public int getId(){
19         return id;
20     }
21
22     public abstract void landOnField(Player player);
23
24 }
25
```

## Fleet.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public class Fleet extends Ownable {
6
7     private final int[] RENTS = new int[]{25,50,100,200};
8
9     public Fleet(int id, Color color, int price) {
10         super(id, color, price);
11     }
12
13     @Override
14     public int getRent() {
15         int ownedFleets = 0;
16         int rent = 0;
17
18         if (this.getOwner() != null){
19             for (Ownable i : this.getOwner().getAccount().getOwnedFields()){
20
21                 if (i instanceof Fleet){
22                     ownedFleets++;
23                 }
24             }
25             rent = RENTS[ownedFleets-1];
26         }
27
28         return rent;
29     }
30 }
31
32 }
33
```



## GoToPrison.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public class GoToPrison extends Field{
6
7     public GoToPrison(int id, Color color) {
8         super(id, color);
9     }
10 }
11 @Override
12 public void landOnField(Player player) {
13     player.getPiece().setPosition(11);
14     player.setPrisonCount(3);
15 }
16
17
18 }
19
```

## Messages.java

```
1 package entity;
2
3 public class Messages {
4     private static String[] chanceMessages = {
5         // Cards resulting in new position of piece
6         /*0*/ "Ryk brikken frem til det nærmeste dampskibsselskab og betal ejeren to gange den leje, han ellers er berettiget til.
Hvis selskabet ikke ejes af nogen, kan De købe det af banken.", // Two of these
7         /*1*/ "Tag med Øresundsbåden - Flyt brikken frem, og hvis De passerer >>Start<<, indkassér kr. 200,00.",
8         /*2*/ "Ryk frem til Frederiksberg Allé. Hvis De passerer >>Start<<, indkassér kr. 200,00.",
9         /*3*/ "Ryk frem til Grønningen. Hvis De passerer >>Start<<, indkassér da kr. 200,00.",
10        /*4*/ "Tag ind på Rådhuspladsen.",
11        /*5*/ "Ryk frem til >>Start<<.",
12        /*6*/ "Ryk tre felter tilbage.", // Two of these
13        // Property charges
14        /*7*/ "Ejendomsskatterne er steget, ekstraudgifterne er: kr. 50,00 pr. hus, kr. 125,00 pr. hotel.",
15        /*8*/ "Kul- og kokspriserne er steget, og De skal betale: kr. 25,00 pr. hus, kr. 125,00 pr. hotel.",
16        // Expenses
17        /*9*/ "De har kørt frem for >>Fuld Stop<<. Betal kr. 100,00 i bøde.",
18        /*10*/ "De har anskaffet et nyt dæk til Deres vogn. Indbetal kr. 100,00.",
19        /*11*/ "Betal kr. 75,00 for modtagne 2 kasser øl.",
20        /*12*/ "De har måttet vedtage en parkeringsbøde. Betal kr. 20,00 til banken.",
21        /*13*/ "Betal for vognvask og smøring kr. 10,00.",
22        /*14*/ "De har været en tur i udlandet og haft for mange cigaretter med hjem. - Betal told kr. 20,00.",
23        // Prison
24        /*15*/ "Gå i fængsel. Ryk direkte til fængslet. Selv om De passerer >>Start<<, indkasserer De ikke kr. 200,00.", // Two of
these
25        // Prison mercy
26        /*16*/ "I anledning af Kongens fødselsdag benådes De herved for fængsel. Dette kort kan opbevares, indtil De får brug for
det, eller De kan sælge det.", // Two of these
27        // For the needy
28        /*17*/ "De modtager >>Matador-legatet for værdig trængende<<, stort kr. 2000,00. Ved værdig trængende forstås, at Deres
formue, d.v.s. Deres kontante penge + skøder + bygninger, ikke overstiger kr. 750,00.",
29        // Bonuses
30        /*18*/ "Deres præmieobligation er kommet ud. De modtager kr. 100,00 af banken.",
31        /*19*/ "Værdien af egen avl fra nyttehaven udgør kr. 200,00, som De modtager af banken.",
32        /*20*/ "Efter auktionen på Assistenthuset, hvor De havde pantsat Deres tøj, modtager De ekstra kr. 108,00.",
33        /*21*/ "De har rettidigt afleveret Deres abonnementskort. Depositum kr. 1,00 udbetales Dem af banken.",
34        /*22*/ "Modtag udbytte af Deres aktier: kr. 50,00.",
35        /*23*/ "Manufakturvarerne er blevet billigere og bedre, herved sparer De kr. 50,00, som De modtager af banken.",
36        /*24*/ "Kommunen har eftergivet et kvartals skat, hæv i banken til en glad aften kr. 150,00.",
```

## Messages.java

```
37      /*25*/ "De har solgt Deres gamle klude. Modtag kr. 20,00.",
38      /*26*/ "Grundet på dyrtiden har De fået gageforhøjelse. Modtag kr 25,00.",
39      // Money collector
40      /*27*/ "De har lagt penge ud til sammenskudsgilde. Mærkværdigvis betaler alle straks. Modtag fra hver medspiller kr.
25,00."
41  };
42  private static String[] fieldNames = {
43      "Start" //Field 1
44      , "Rødovrevej" //Field 2
45      , "Prøv lykken" //Field 3
46      , "Hvidovre" //Field 4
47      , "Betal Indkomstskat" //Field 5
48      , "Øresund A/S" //Field 6
49      , "Roskildevej" //Field 7
50      , "Prøv lykken" //Field 8
51      , "Valby Langgade" //Field 9
52      , "Allégade" //Field 10
53      , "Fængsel" //Field 11
54      , "Frederiksberg Allé" //Field 12
55      , "Tuborg" //Field 13
56      , "Bülowsvej" //Field 14
57      , "Gl. Kongevej" //Field 15
58      , "D.F.D.S" //Field 16
59      , "Bernstorffsvej" //Field 17
60      , "Prøv lykken" //Field 18
61      , "Hellerupvej" //Field 19
62      , "Strandvej" //Field 20
63      , "Helle" //Field 21
64      , "Trianglen" //Field 22
65      , "Prøv lykken" //Field 23
66      , "Østerbrogade" //Field 24
67      , "Grønningen" //Field 25
68      , "Ø.K." //Field 26
69      , "Bredgade" //Field 27
70      , "Kgs. Nytorv" //Field 28
71      , "Carlsberg" //Field 29
72      , "Østergade" //Field 30
73      , "De sættes i fængsel" //Field 31
74      , "Amagertorv" //Field 32
75      , "Vimmelskaftet" //Field 33
```

# Messages.java

```

76         , "Prøv lykken"           //Field 34
77         , "Nygade"               //Field 35
78         , "D/S Bornholm"         //Field 36
79         , "Prøv lykken"           //Field 37
80         , "Frederiksberggade"     //Field 38
81         , "Ekstraordinær Statsskat" //Field 39
82         , "Rådhuspladsen"         //Field 40
83     };
84     private static String[] boardMessages = {
85         "Pris:",                   //0
86         "Leje:",                   //1
87         "Modtag:",                 //2
88         "Betal:",                  //3
89         "eller",                   //4
90         "af alle ejendele",        //5
91     };
92
93     private static String[] generalMessages = {
94         /*0*/ "Denne ejendom er ikke ejet af nogen spiller. Vil De købe den for ",
95         /*1*/ "Ja",
96         /*2*/ "Nej",
97         /*3*/ "De har nu to muligheder",
98         /*4*/ "Betal ",
99         /*5*/ "% af alle ejendele ",
100        /*6*/ "Vil De starte et nyt spil?",
101        /*7*/ "Slå med terningerne",
102        /*8*/ "Hvor mange spillere skal deltage i spillet?",
103        /*9*/ "De er landet på en anden spillers ejendom. De skal betale ",
104        /*10*/ "Spiller ",
105        /*11*/ "Det er ",
106        /*12*/ "'s tur.",
107        /*13*/ "OK",
108        /*14*/ "Tillykke ",
109        /*15*/ ", De har vundet spillet!",
110        /*16*/ " i leje.",
111        /*17*/ "De skal betale ",
112        /*18*/ " til skattefar.",
113        /*19*/ "De modtager ",
114        /*20*/ "De er landet på deres egen ejendom og nyder de dejlige omgivelser.",
115        /*21*/ "Køb hus eller hotel",

```

Messages.java

```
116         /*22*/ "Leje: ",
117         /*23*/ "De skal betale ",
118         /*24*/ " kr. Er De sikker?",
119         /*25*/ "De har ikke nok penge til at købe dette felt.",
120         /*26*/ "De er landet på ",
121         /*27*/ "De har passeret ",
122         /*28*/ ", og modtager ",
123         /*29*/ "De er blevet stoppet af politiet og sendes direkte i fængsel!",
124         /*30*/ "Hvilket felt vil De købe hus eller hotel til?",
125         /*31*/ "Betal 50 kr.",
126         /*32*/ "De er i fængsel. For at komme ud kan de betale 50 kr. eller slå to ens"
127     };
128
129     public static String[] getChanceMessages(){
130         return chanceMessages;
131     }
132
133     public static String[] getFieldNames(){
134         return fieldNames;
135     }
136
137     public static String[] getBoardMessages(){
138         return boardMessages;
139     }
140
141     public static String[] getGeneralMessages(){
142         return generalMessages;
143     }
144 }
145
```

## Ownable.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public abstract class Ownable extends Field{
6
7     protected int price;
8     protected Player owner;
9
10    public Ownable(int id, Color color, int price){
11        super(id, color);
12        this.price = price;
13    }
14
15    @Override
16    public void landOnField(Player player){
17        int balance = player.getAccount().getBalance();
18
19        //no owner of field
20        if (owner == null){
21            //player can afford field and wants to buy field
22            if (balance >= price && player.getChoice().equals(Messages.getGeneralMessages()[1])){
23                owner = player;
24                player.getAccount().setOwnedField(this); //Field is added to Player's owned fields
25                player.getAccount().setBalance(balance-price); //Balance of Player is changed
26            }
27        }
28        //player owns field
29        else if (owner == player){
30
31        }
32        //pay rent to owner if he is not bankrupt
33        else if (owner.getAccount().getBalance() >= 0){
34            int rent = 0;
35            if (this instanceof Brewery){
36                //Brewery is a special case as we need the player's last dice throw to calculate rent
37                //Since we do not pass a player to getRent(), we need to do this multiplication on this level
38                rent = getRent()*player.getLastThrow().getSum();
39            }
40            else{
```

## Ownable.java

```
41         rent = getRent();
42     }
43     //if the player does not have enough money to pay all the rent, the owner should only be given what's left
44     if(player.getAccount().getBalance() < rent){
45         owner.getAccount().setBalance(owner.getAccount().getBalance() + player.getAccount().getBalance());
46     }
47     else{
48         owner.getAccount().setBalance(owner.getAccount().getBalance() + rent);
49     }
50     player.getAccount().setBalance(balance - rent);
51 }
52 }
53
54 public int getPrice(){
55     return price;
56 }
57
58 public abstract int getRent();
59
60 public Player getOwner(){
61     return owner;
62 }
63
64 public void setOwner(Player owner){
65     this.owner = owner;
66 }
67
68 }
69
```

## Piece.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public class Piece {
6     private int position;
7     private Color color;
8
9     // Constructor - In order to create the piece, you will need to give the vehicle a color using java.awt.Color;
10    public Piece(Color color) {
11        this.color = color;
12        position = 1; //player starts on field 1
13    }
14
15    public Color getColor() {
16        return this.color;
17    }
18
19    // Move the piece to a position using an integer
20    public void setPosition(int position) {
21        this.position = position;
22    }
23
24    public int getPosition() {
25        return position;
26    }
27
28 }
29
```



## Player.java

```
1 package entity;
2
3 public class Player {
4
5     private int id;
6     private String name;
7     private Piece piece;
8     private Account account;
9     private String choice;
10    private int prisonCount; //counts number of turns in prison. If 0 then player is not in prison.
11    private int equalCount; //counts number of equal diceValues in DiceCup in a row.
12    private DiceCup lastThrow;
13
14    public Player(int id, String name, Piece piece, Account account){
15        this.id = id;
16        this.name = name;
17        this.piece = piece;
18        this.account = account;
19        prisonCount = 0;
20        equalCount = 0;
21    }
22
23    public String getName(){
24        return name;
25    }
26
27    public Account getAccount(){
28        return account;
29    }
30
31    public Piece getPiece(){
32        return piece;
33    }
34
35    public int getID(){
36        return id;
37    }
38
39    public String getChoice(){
40        return choice;
41    }
42 }
```

Player.java

```
41  }
42
43  public void setChoice(String choice){
44      this.choice = choice;
45  }
46
47  public int getPrisonCount(){
48      return prisonCount;
49  }
50
51  public void setPrisonCount(int prisonCount){
52      this.prisonCount = prisonCount;
53  }
54
55  public int getEqualCount(){
56      return equalCount;
57  }
58
59  public void setEqualCount(int equalCount){
60      this.equalCount = equalCount;
61  }
62
63  public DiceCup getLastThrow(){
64      return lastThrow;
65  }
66
67  public void setLastThrow(DiceCup lastThrow){
68      this.lastThrow = lastThrow;
69  }
70
71
72 }
73
```

## Refuge.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public class Refuge extends Field{
6
7     public Refuge(int id, Color color) {
8         super(id, color);
9     }
10 }
11
12 @Override
13 public void landOnField(Player player) {
14
15 }
16
17 }
18
```

## Street.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public class Street extends Ownable {
6
7     private final int[] rents;
8     private int housesOwned;
9     private final int housePrice;
10
11     public Street(int id, Color color, int price, int[] rents, int housePrice) {
12         super(id, color, price);
13
14         this.rents = rents;
15         housesOwned = 0;
16         this.housePrice = housePrice;
17     }
18
19     @Override
20     public int getRent() {
21         int rent = 0;
22
23         if (housesOwned == 0 && owner != null){
24             //find amount of streets in group
25             int groupAmount;
26             if (color == Color.cyan || color == Color.orange){
27                 groupAmount = 2;
28             }
29             else{
30                 groupAmount = 3;
31             }
32
33             //find streets in group owned by player
34             int ownedInGroup = 0;
35             for(Ownable i : owner.getAccount().getOwnedFields()){
36                 if (i.getColor() == color && i instanceof Street){
37                     ownedInGroup++;
38                 }
39             }
40         }
```

Street.java

```
41         //double rent
42         if (ownedInGroup == groupAmount){
43             rent = rents[0]*2;
44         }
45         else{
46             rent = rents[0];
47         }
48     }
49     else{
50         rent = rents[housesOwned];
51     }
52
53     return rent;
54 }
55
56
57 public int getHousesOwned(){
58     return housesOwned;
59 }
60
61 public int getHousePrice(){
62     return housePrice;
63 }
64
65 public void setHousesOwned(int amount){
66     housesOwned = amount;
67 }
68
69 }
70
```

## Tax.java

```
1 package entity;
2
3 import java.awt.Color;
4
5 public class Tax extends Field {
6     private int taxRate; //in percent. If 0 then no taxRate
7     private int taxAmount;
8
9     public Tax(int id, Color color, int taxRate, int taxAmount){
10         super(id, color);
11         this.taxRate = taxRate;
12         this.taxAmount = taxAmount;
13     }
14
15     @Override
16     public void landOnField(Player player){
17         int balance = player.getAccount().getBalance();
18
19         if (taxRate > 0){
20             String choice = player.getChoice();
21             if (choice.equals(Messages.getGeneralMessages()[4] + taxAmount)){//user chooses taxAmount
22                 player.getAccount().setBalance(balance - taxAmount);
23             }
24             else{//user chooses taxRate
25                 player.getAccount().setBalance(balance - (int)((taxRate/100.0) * player.getAccount().calculateAssets()));
26             }
27         }
28         else{
29             player.getAccount().setBalance(balance - taxAmount);
30         }
31     }
32
33     public int getTaxAmount(){
34         return taxAmount;
35     }
36
37     public int getTaxRate(){
38         return taxRate;
39     }
40 }
```