```java
package game;

import game.Account.IllegalAmountException;

public class BoardController {

    private static Board board;
    //Constructor
    public BoardController(int numberOfFields) {
        board = new Board(numberOfFields);
        board.setupDefaultBoard();
    }
    //Overload - ordinary landOnField
    public void landOnField(Player player, int fieldNumber, Decorator decorator,
            PlayerController playerController, CardController cardController){
        landOnField(player, fieldNumber, decorator, playerController, cardController, 1);
    }

    public void landOnField(Player player, int fieldNumber, Decorator decorator,
            PlayerController playerController, CardController cardController, int rentModifier){
        //finds the field the player landed on
        Field field = board.getFieldByNumber(fieldNumber);
        //Checks fieldType -(Pseudo polymorphy)
        if (field instanceof Ownable){
            landOnOwnable(player, decorator, playerController, rentModifier, field);
        }
        if (field instanceof Chance){
            System.out.println("" + player + decorator + playerController + this);
            //Delegate responsibility to CardController
            cardController.drawCard(player, decorator, playerController, this);
        }
        if (field instanceof Refuge){
            decorator.showMessage(((Refuge)field).decoratorMessage(player));
        }
        if (field instanceof GotoJail){
            decorator.showMessage(((GotoJail)field).decoratorMessage(player));
            playerController.moveToJail(player, getJailNumber());
        }
        if (field instanceof Tax){
            landOnTax(player, decorator, playerController, field);
        }
```

```
45          decorator.updatePlayer(player);
46      }
47
48      private void landOnOwnable(Player player, Decorator decorator,
49              PlayerController playerController, int rentModifier, Field
  field) {
50          Player owner = ((Ownable) field).getOwner();
51          if (owner == null){
52              //Unowned Field
53              offerOwnableToPlayer((Ownable) field, player, decorator,
  playerController);
54          } else if (owner != player){
55              System.out.println("Started code for hostile field");
56              //Owned by other pLayer
57              String[] msg = new String[]
  {"OwnedByOtherPlayer",owner.getPlayerName(),"YouMustPayRent",String.valueOf(
  ((Ownable) field).getRent())};
58              int rent = ((Ownable)field).getRent()*rentModifier;
59              //Raise money
60              boolean rentPayed = playerController.payDebt(player, owner,
  decorator, msg, rent);
61              if(!rentPayed){
62                  String[] msg2 = new String[]{"YouAreBankrupt"};
63                  decorator.showMessage(msg2);
64                  playerController.hostileTakeOver(owner, player);
65              }
66              //update players
67              decorator.updatePlayer(player);
68              decorator.updatePlayer(owner);
69          } else if (owner == player){
70              String[] msg = new String[] {"OwnField"};
71              decorator.showMessage(msg);
72          }
73      }
74
75
76      private void landOnTax(Player player, Decorator decorator,
77              PlayerController playerController, Field field) {
78          int taxAmount = ((Tax) field).getTaxAmount();
79          int taxRate = ((Tax)field).getTaxrate();
80          boolean debtPayed = false;
81          //Determine TaxAmount
82          if (taxRate<0){
83              //Field is fixed tax amount
84              decorator.showMessage(((Tax)field).decoratorMessage(player));
85          } else {
86              //Select taxAmount or Tax rate
87              String[] options = new String[]{String.valueOf(taxAmount),
  String.valueOf(taxRate)+"%"};
88              int userSelection =
```

```
   decorator.getUserSelection((((Tax)field).decoratorMessage(player)),
   options);
 89            if (userSelection == 1){
 90                taxAmount = (int)( ((taxRate/100.f)*
   playerController.getTotalAssets(this, player)));
 91                System.out.println("Valgt 10%" + taxAmount);
 92            }
 93        }
 94        String[] msg0 = new String[]
   {"YouMustPayTax",String.valueOf(taxAmount)};
 95        debtPayed = playerController.payDebt(player, null, decorator, msg0,
   taxAmount);
 96        //Pay debt
 97        if (!debtPayed) {
 98            String[] msg2 = new String[]{"YouAreBankrupt"};
 99            decorator.showMessage(msg2);
100        }
101        decorator.updatePlayer(player);
102    }
103
104    private void offerOwnableToPlayer(Ownable ownable, Player player,
105            Decorator decorator, PlayerController playerController) {
106        String[] Fieldmsg = ownable.decoratorMessage(player);
107        String[] msg = new String[]{"OfferFieldToPlayer"};
108        msg = StringTools.add(Fieldmsg,msg);
109
110        boolean pricePayed = false;
111        while (!pricePayed){
112            String[] opt0 = new String[]{"Yes", "No"};
113            int userChoice = decorator.getUserButtonPressed(msg, opt0);
114            if(userChoice == 0){
115                try {
116                    player.getAccount().withdraw(ownable.getPrice());
117                    ownable.setOwner(player);
118                    decorator.updateFieldOwner(ownable);
119                    if (ownable instanceof Shipping){
120                        for (Field field :
   BoardController.getBoard().getFields()){
121                            // to make sure all fleets are updated
122                            decorator.updateFieldRent(field);
123                        }
124                    }
125                    decorator.updatePlayer(player);
126                    pricePayed = true;
127                } catch (InsufficientFundsException e) {
128
129                    if(getFieldsbyPlayer(player).length == 0){
130                        String[] msg0 = new String[]{"NoFieldsToTrade"};
131                        decorator.showMessage(msg0);
132                        break;
```

```
133                     }
134                     else
    if(!playerController.handleInsufficientFunds(player, ownable.getPrice(),
    decorator)){
135                         break;
136                     }
137                 } catch (IllegalAmountException e) {
138                     System.err.println("Fail in offerOwnableToPlayer");
139                     e.printStackTrace();
140                 }
141             }
142             else break;
143         }
144     }
145
146
147
148
149     // Static helper Method to get array of a players owned fields
150     public static Ownable[] getFieldsbyPlayer(Player player) {
151
152         int numberOfOwnedFields = 0;
153
154         // counts number of players owned fields
155         for (int i=0; i<board.getFields().length; i++){
156             Field field = board.getFields()[i];
157             if(field instanceof Ownable){
158                 if(((Ownable)field).getOwner() == player){
159                     numberOfOwnedFields++;
160                 }
161             }
162         }
163         int j =0;
164         Ownable[] ownedFieldsByPlayer = new Ownable[numberOfOwnedFields];
165         for (int i =0;i < board.getFields().length ;i++){
166             Field field = board.getFields()[i];
167             if (field instanceof Ownable){
168                 if (((Ownable) field).getOwner()== player){
169                     ownedFieldsByPlayer[j] = ((Ownable)field);
170                     j++;
171                 }
172             }
173         }
174         return ownedFieldsByPlayer;
175     }
176
177
178     public static int getFieldNumber(Field field){
179         int fieldNumber = 0;
180         for(int i = 0 ; i < board.getFields().length; i++){
```

```
181             if(field.equals(board.getFields()[i])){
182                 fieldNumber = i+1;
183             }
184         }
185         return fieldNumber;
186     }
187 
188     public static boolean hasAnyUnPawnedFields(Player player){
189 
190         boolean unPawnedFields = false;
191         Field[] ownedFields = getFieldsbyPlayer(player);
192         for(Field ownable: ownedFields){
193 
194             if(ownable instanceof Ownable && !
    ((Ownable)ownable).isPawned()){
195                 unPawnedFields = true;
196                 break;  // breaks for loop if only one field is not pawned
197             }
198             else unPawnedFields = false;
199 
200         }
201         return unPawnedFields;
202     }
203 
204 
205     public static Board getBoard() {
206         return board;
207     }
208 
209     //
210     //  private static void setBoard(Board board) {
211     //      BoardController.board = board;
212     //  }
213     //
214 
215     public static int getNumberOfFleets(Player player) {
216         int numberOfFleets = 0;
217         Field[] fields = getFieldsbyPlayer(player);
218         for (Field field : fields)
219             if (field instanceof Shipping){
220                 numberOfFleets++;
221             }
222 
223         return numberOfFleets;
224     }
225     public int getJailNumber() {
226 
227         return getFieldNumber(getJail());
228     }
229     public Jail getJail() {
```

```
230         Jail jail = null;
231         for (Field field : board.getFields()){
232             if (field instanceof Jail) jail = (Jail) field;
233         }       return jail;
234     }
235
236 }
237
```