```java
package game;
import java.awt.Color;

public class Decorator {
    private static final int START_FIELD = 1;
    private Translator translator;

    //Constructor
    public Decorator(String language) {
        super();
        this.translator = new Translator(language);
    }

    //Setup GUI
    public void setupGUI(game.fields.Field[] gameFields, game.Player[] gamePlayers ) {
        GUI.create("Fields.txt");
        //add players to GUI
        if (gamePlayers!=null) {
            for (int i = 0; i < gamePlayers.length; i++) {
                if (gamePlayers[i].getPlayerName().equals("Casper")){
                    GUI.addPlayer(gamePlayers[i].getPlayerName(),
                            gamePlayers[i].getAccount().getBalance(), new Color(0, 0, 255, 64));
                }else {
                    GUI.addPlayer(gamePlayers[i].getPlayerName(), gamePlayers[i]
                            .getAccount().getBalance(),getColor(i));
                    GUI.setCar(START_FIELD, gamePlayers[i].getPlayerName());
                }
            }
        }
        //Update Fields to new prices and texts
        for (int j = 0; j<gameFields.length;j++){
            if (gameFields[j] instanceof game.fields.Ownable){
                String[] priceText = new String[] {"Price", ": "};
                GUI.setSubText(j+1, translator.translateConcat(priceText) + ((game.fields.Ownable) gameFields[j]).getPrice());
            }
            if (gameFields[j] instanceof game.fields.Refuge){
                GUI.setSubText(j+1, translator.translateConcat(((game.fields.Refuge)gameFields[j]).getSubText()));
                GUI.setDescriptionText(j+1, translator.translateConcat(((game.fields.Refuge)gameFields[j]).getDescription()));
            }
        }
    }
```

```
45      private Color getColor(int i) {
46          switch (i) {
47          case 0:
48              return Color.BLUE;
49          case 1:
50              return Color.RED;
51          case 2:
52              return Color.GREEN;
53          case 3:
54              return Color.YELLOW;
55          case 4:
56              return Color.BLACK;
57          case 5:
58              return Color.WHITE;
59          default:
60              break;
61          }
62          return Color.PINK;
63      }
64
65      //ordinary methods
66      //Show Message
67      public void showMessage(String[] msg) {
68          GUI.showMessage(translator.translateConcat(msg));
69      }
70      //Get user input string
71      public String getUserString(String[] messageString) {
72          return
    GUI.getUserString(translator.translateConcat(messageString));
73      }
74      //get userSelectionnumber from dropdown box
75      public int getUserSelection(String[] msg, String[] options) {
76          String[] translatedOptions = translator.translate(options);
77          String translatedMsg = translator.translateConcat(msg);
78          String selectedOption = GUI.getUserSelection(translatedMsg,
    translatedOptions);
79          int selectionNumber = 0;
80          for (int i = 0;i<options.length;i++){
81              if (selectedOption == translatedOptions[i])
82                  selectionNumber = i;
83          }
84          return selectionNumber;
85      }
86      // Get number of button Pressed
87      public int getUserButtonPressed(String[] messageString, String[]
    buttons) {
88          String[] translatedButtons = translator.translate(buttons);
89          String translatedMsg = translator.translateConcat(messageString);
90          String selectedButton = GUI.getUserButtonPressed(translatedMsg,
    translatedButtons);
```

```
91          int buttonNumber = 0;
92          for (int i = 0;i<buttons.length;i++){
93              if (selectedButton == translatedButtons[i])
94                  buttonNumber = i;
95          }
96          return buttonNumber;
97          //Show dice - from array of ints
98      }
99      public void updateDice(DiceCup diceCup) {
100         GUI.setDice(diceCup.getDiceFaceValues()[0],
    diceCup.getDiceFaceValues()[1]);
101     }
102
103     public void updatePlayer(game.Player player){
104         updatePlayerField(player);
105         updatePlayerBalance(player);
106         if (player.isBroke()==true)
    GUI.removeAllCars(player.getPlayerName());
107     }
108     //Move player to Field
109     private void updatePlayerField(game.Player player) {
110         GUI.removeAllCars(player.getPlayerName());//Get rid of unwanted
    cars
111         GUI.setCar(player.getCurrentFieldNumber(),
    player.getPlayerName());//setCar for player
112     }
113     //Update Players Balance
114     private void updatePlayerBalance(game.Player player) {
115         GUI.setBalance(player.getPlayerName(),
    player.getAccount().getBalance());
116     }
117
118     //Update Fieldrent
119     public void updateFieldRent(game.fields.Field field) {
120         if (field instanceof game.fields.Ownable){
121             String[] rentText = new String[] {"Rent", ": ",
    String.valueOf(((game.fields.Ownable) field).getRent())};
122             GUI.setLeje(BoardController.getFieldNumber(field),
    translator.translateConcat(rentText) );
123         }
124     }
125
126     //Remove field Rent Text
127     public void removeRent(game.fields.Field field){
128         GUI.setLeje(BoardController.getFieldNumber(field), "");
129     }
130     //Update Houses
131     public void updateHouses(game.fields.Field field){
132         if (field instanceof game.fields.Street){
133             if (((game.fields.Street)field).getBuildings()<5 &&
```

```
    ((game.fields.Street)field).getBuildings()>=0){
134                 GUI.setHotel(BoardController.getFieldNumber(field), false);
135                 GUI.setHouses(BoardController.getFieldNumber(field),
    ((game.fields.Street) field).getBuildings());
136             } else if (((game.fields.Street)field).getBuildings()==5){
137                 GUI.setHouses(BoardController.getFieldNumber(field), 0);
138                 GUI.setHotel(BoardController.getFieldNumber(field), true);
139             } else {
140                 System.err.println("Illegal HouseCount - updateHouses -
    decorator");
141             }
142             updateFieldRent(field);
143         }
144     }
145     //Set Field owner
146     public void updateFieldOwner(game.fields.Ownable field) {
147         System.out.println(BoardController.getFieldNumber(field));
148         if(field.getOwner()==null) {
149             GUI.removeOwner(BoardController.getFieldNumber(field));;
150         } else {
151             System.out.println(field.getOwner().getPlayerName());
152             GUI.setOwner(BoardController.getFieldNumber(field),
    field.getOwner().getPlayerName());
153         }
154         updateFieldRent(field);
155     }
156
157     public void updateFieldPrice(game.fields.Field field) {
158         String[] text = new String[] {"Price", ": ",
    String.valueOf(((game.fields.Ownable)field).getPrice())};
159         GUI.setSubText(BoardController.getFieldNumber(field),
    translator.translateConcat(text));
160     }
161     public void updatePawned (game.fields.Ownable ownable){
162         String title = ownable.isPawned() ? ownable.getTitle() + " " +
    "Pawned" : ownable.getTitle();
163         System.out.println(title);
164         GUI.setTitleText(BoardController.getFieldNumber(ownable), title);
165     }
166
167     public void showChanceCard(String[] chanceText){
168         String msg = translator.translateConcat(chanceText);
169         GUI.displayChanceCard(msg);
170     }
171
172     //testDriver
173     public static void main(String[] args){
174         Decorator testDecorator = new Decorator("danish");
175         @SuppressWarnings("unused")
176         BoardController testbc = new BoardController(40);
```

```
177         game.fields.Field[] testFields =
    BoardController.getBoard().getFields();
178         game.fields.Field testField = testFields[1];
179         Player[] testPlayers = new Player[] {new Player("Casper", 400000),
    new Player("Frans",300000)};
180         testDecorator.setupGUI(testFields, testPlayers);
181         testPlayers[0].setCurrentFieldNumber(5);
182         testDecorator.updatePlayerField(testPlayers[0]);//Works
183         try {testPlayers[0].getAccount().deposit(10000000);
184         } catch (Exception e) { }
185         testDecorator.updatePlayerBalance(testPlayers[0]);;//Works
186         System.out.println(testFields[1]+""+testPlayers[0]);
187
188         ((Ownable) testField).setOwner(testPlayers[0]);
189         testDecorator.updateFieldOwner((Ownable) testFields[1]);
190         //String[] testStringArray = {"BuyOut", "RollDice"};
191         ((Ownable)testField).setOwner(null);
192         testDecorator.updateFieldOwner((Ownable) testField);
193         System.out.println(((game.fields.Ownable)testFields[1]).getRent());
194         ((game.fields.Street)testField).addBuilding();
195         testDecorator.updateHouses(testField);
196         //((game.fields.Street)testField).setBuildings(5);
197         testDecorator.updateHouses(testField);
198         //((game.fields.Street)testField).setBuildings(0);
199         testDecorator.updateHouses(testField);
200         //((game.fields.Street)testField).setBuildings(6);
201         testDecorator.updateHouses(testField);
202         //((game.fields.Street)testField).setBuildings(-1);
203         testDecorator.updateHouses(testField);
204         ((game.fields.Ownable)testField).setPawned(true);
205         testDecorator.updatePawned((Ownable) testField);
206         //      System.out.println(testDecorator.getUserButtonPressed("test
    Buttons", testStringArray));//Works Like a Charm
207         //      //System.out.println(testDecorator.getUserSelection("testTe
    xt", testStringArray));//Works
208     }
209
210
211
212 }
213
```