File - E:\Projects\newBoardGameProject\src\main\java\com\example\template\Player.java

```java
package com.example.template;

import java.util.ArrayList;

public class Player {
    private String name;
    private int amountOfTerritories = 0;
    private ArrayList<Territory> allTerritoriesControlled = new ArrayList<>();
    private ArrayList<Territory> attackTerritories = new ArrayList<>();
    String color;

    public Player(String name, int amountOfTerritories, Territory startingTerritory, String color) {
        this.name = name;
        this.amountOfTerritories = amountOfTerritories;
        allTerritoriesControlled.add(startingTerritory);
        this.color = color;
    }

    public void addToAttackTerritories(Territory territory) {
        attackTerritories.add(territory);
    }

    public ArrayList<Territory> getAttackTerritories() {
        return attackTerritories;
    }

    public Player(String color) {
        this.color = color;
    }

    public ArrayList<Territory> getAllTerritoriesControlled() {
        return allTerritoriesControlled;
```

```
33      }
34
35      public String getName() {
36          return name;
37      }
38
39      public int getAmountOfTerritories() {
40          return amountOfTerritories;
41      }
42
43      public String getColor() {
44          return color;
45      }
46  }
47
```

```java
package com.example.template;

import java.util.ArrayList;

public class Territory {
    private String territoryType;
    private int territoryTypeID;
    private Player controllingPlayer;
    private Coordinates coordOfTerritory;
    private int amountOfSoldiers;

    public Territory(int row, int column, int amountOfSoldiers, int territoryTypeID) {
        coordOfTerritory = new Coordinates(row,column);
        this.amountOfSoldiers = amountOfSoldiers;
        this.territoryTypeID = territoryTypeID;
    }

    public Territory(int row, int column, Player controllingPlayer, int amountOfSoldiers,int territoryTypeID) {
        coordOfTerritory = new Coordinates(row,column);
        this.controllingPlayer = controllingPlayer;
        this.amountOfSoldiers = amountOfSoldiers;
        this.territoryTypeID = territoryTypeID;
    }

    public int getTerritoryTypeID() {
        return territoryTypeID;
    }

    public int getAmountOfSoldiers() {
        return amountOfSoldiers;
    }

    public Coordinates getCoordOfTerritory() {
        return coordOfTerritory;
    }

```

```
37      public void setControllingPlayer(Player
    controllingPlayer) {
38          this.controllingPlayer = controllingPlayer;
39      }
40
41      public Player getControllingPlayer() {
42          return controllingPlayer;
43      }
44  }
45
```

```java
package com.example.template;

public class Coordinates {
    private int row;
    private int column;
//    private boolean isABorderCord = false;

    public Coordinates(int row, int column) {
        this.row = row;
        this.column = column;
    }

    public int getRow() {
        return row;
    }

    public int getColumn() {
        return column;
    }
}

```

```java
package com.example.template;

import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Border;
import javafx.scene.layout.GridPane;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;

import javax.swing.JOptionPane;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.PrintWriter;
import java.lang.reflect.Array;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Objects;
import java.util.Scanner;

public class HelloController {
    @FXML
    public GridPane mainGridPane;
    @FXML
    public Button startButton;
    @FXML
    public Label attacksLeftLabel;
    @FXML
    public Label MainDisplayLabel;
    @FXML
    public Button turnDisplayButton;
    @FXML
    public Label currentTurnLabel;
    @FXML
    public Label creditsLabel;
    @FXML
```

```
42      public CheckBox possibleOptionsCheckBox;
43      @FXML
44      public Rectangle blurRectangle;
45      @FXML
46      public GridPane secondaryGridPane;
47      @FXML
48      public Rectangle menuRectangle;
49      @FXML
50      public Label operationBoxLabel3;
51      @FXML
52      public Label operationBoxLabel2;
53      @FXML
54      public GridPane AttackGridPane;
55      @FXML
56      public Label operationBoxLabel1;
57      @FXML
58      public Label operationBoxHeaderLabel;
59      @FXML
60      public Button operationBoxCancelButton;
61      @FXML
62      public Button operationBoxAcceptButton;
63      @FXML
64      public AnchorPane operationBOX;
65
66
67      //    @FXML
68  //    public ImageView tempImageView;
69      Button[][] buttonArray = new Button[10][15];
70      Button[][] attackButtonArray = new Button[3][3];
71      Button[][] displayButtonArray = new Button[10][15
    ];
72      ImageView[][] imageViewArray = new ImageView[10][
    15];
73      Territory[][] territoryArray = new Territory[10][
    15];
74      private int numPlayers;
75      private int turn;
76      private ArrayList<Player> allPlayers = new
    ArrayList<>();
77      private boolean editingMode = false;
78      private int attacksLeft = 3;
```

```
 79     private int row;
 80     private int column;
 81
 82     public void start() {
 83         mainGridPane.setVisible(true);
 84         startButton.setVisible(false);
 85         startButton.setDisable(true);
 86         for(int row =0;row<buttonArray.length;row
    ++) {
 87             for(int column=0;column<buttonArray[0].
    length;column++) {
 88                 territoryArray[row][column] = new
    Territory(row,column, new Player("-fx-background-
    color: LightBlue;"),5,0);
 89                 buttonArray[row][column] = new
    Button("");
 90                 buttonArray[row][column].
    setPrefHeight(100);
 91                 buttonArray[row][column].
    setPrefWidth(100);
 92                 displayButtonArray[row][column] =
    new Button("");
 93                 displayButtonArray[row][column].
    setPrefHeight(100);
 94                 displayButtonArray[row][column].
    setPrefWidth(100);
 95                 imageViewArray[row][column] = new
    ImageView();
 96                 imageViewArray[row][column].
    setFitHeight(50);
 97                 imageViewArray[row][column].
    setFitWidth(50);
 98 //                 Font font = new Font(0);
 99 //                 buttonArray[row][column].setFont(
    font);
100                 buttonArray[row][column].setStyle("-
    fx-border-color: LightBlue; -fx-background-color:
    LightBlue; -fx-border-radius: 0px; -fx-background-
    radius: 0px; ");
101                 displayButtonArray[row][column].
    setStyle("-fx-border-color: transparent; -fx-
```

```
101 background-color: transparent; -fx-border-radius:
    0px; ");
102                 mainGridPane.add(buttonArray[row][
    column],column,row);
103                 secondaryGridPane.add(
    displayButtonArray[row][column],column,row);
104             }
105         }
106         askForAmountOfPlayers();
107         setUpPlayers();
108         turn = (int) (Math.random()*numPlayers);
109         playTurn();
110         //printWorldBorders();
111         EventHandler z = new EventHandler<
    ActionEvent>() {
112             @Override
113             public void handle(ActionEvent event) {
114                 row = GridPane.getRowIndex((Button)
    event.getSource());
115                 column = GridPane.getColumnIndex((
    Button) event.getSource());
116                 if (editingMode) {
117                     if (validTerritory(row, column,
    true)) {
118                         System.out.println("start
    editing that boi");
119                     }
120                     else {
121                         System.out.println("You cant
     edit that territory!");
122                     }
123                 }
124                 else {
125                     if (validTerritory(row, column,
    false)) {
126                         setUpAttackUI();
127 //                        territoryArray[row][column
    ].setControllingPlayer(allPlayers.get(turn%
    numPlayers));
128 //                        allPlayers.get(turn%
    numPlayers).getAllTerritoriesControlled().add(
```

```
128 territoryArray[row][column]);
129 //                          updateMap();
130 //                          generateAreasToAttack();
131                     }
132                     else {
133                         System.out.println("you cant
     attack that!!!");
134                     }
135                 }
136             }
137         };
138         for(int row =0;row<buttonArray.length;row
    ++) {
139             for(int column=0;column<buttonArray[0].
    length;column++) {
140                 buttonArray[row][column].setOnAction
    (z);
141             }
142         }
143     }
144
145     private boolean firstTime = true;
146     private Boolean[][] activatedButtons = new
    Boolean[3][3];
147     public void setUpAttackUI() {
148         openAttackPlan();
149         if (firstTime) {
150             firstTime = false;
151             setUpAttackGridPane();
152         }
153         for(int row =0;row<3;row++) {
154             for(int column=0;column<3;column++) {
155                 activatedButtons[row][column] =
    false;
156                 attackButtonArray[row][column].
    setDisable(false);
157             }
158         }
159         displayAttackGridPane();
160         tallyUpTotals();
161     }
```

```
162
163     public void displayAttackGridPane() {
164         ArrayList<Boolean> directionsCanGo =
    findDirections(territoryArray[row][column]);
165         if (directionsCanGo.get(0)) {
166             attackButtonArray[0][1].setStyle(
    buttonArray[row-1][column].getStyle());
167         }
168         else {
169             attackButtonArray[0][1].setStyle("-fx-
    border-color: Black; -fx-background-color: Black;");
170             attackButtonArray[0][1].setDisable(true
    );
171         }
172         if (directionsCanGo.get(1)) {
173             attackButtonArray[0][2].setStyle(
    buttonArray[row-1][column+1].getStyle());
174         }
175         else {
176             attackButtonArray[0][2].setStyle("-fx-
    border-color: Black; -fx-background-color: Black;");
177             attackButtonArray[0][2].setDisable(true
    );
178         }
179         if (directionsCanGo.get(2)) {
180             attackButtonArray[1][2].setStyle(
    buttonArray[row][column+1].getStyle());
181         }
182         else {
183             attackButtonArray[1][2].setStyle("-fx-
    border-color: Black; -fx-background-color: Black;");
184             attackButtonArray[1][2].setDisable(true
    );
185         }
186         if (directionsCanGo.get(3)) {
187             attackButtonArray[2][2].setStyle(
    buttonArray[row+1][column+1].getStyle());
188         }
189         else {
190             attackButtonArray[2][2].setStyle("-fx-
    border-color: Black; -fx-background-color: Black;");
```

```
191             attackButtonArray[2][2].setDisable(true
    );
192         }
193         if (directionsCanGo.get(4)) {
194             attackButtonArray[2][1].setStyle(
    buttonArray[row+1][column].getStyle());
195         }
196         else {
197             attackButtonArray[2][1].setStyle("-fx-
    border-color: Black; -fx-background-color: Black;");
198             attackButtonArray[2][1].setDisable(true
    );
199         }
200         if (directionsCanGo.get(5)) {
201             attackButtonArray[2][0].setStyle(
    buttonArray[row+1][column-1].getStyle());
202         }
203         else {
204             attackButtonArray[2][0].setStyle("-fx-
    border-color: Black; -fx-background-color: Black;");
205             attackButtonArray[2][0].setDisable(true
    );
206         }
207         if (directionsCanGo.get(6)) {
208             attackButtonArray[1][0].setStyle(
    buttonArray[row][column-1].getStyle());
209         }
210         else {
211             attackButtonArray[1][0].setStyle("-fx-
    border-color: Black; -fx-background-color: Black;");
212             attackButtonArray[1][0].setDisable(true
    );
213         }
214         if (directionsCanGo.get(7)) {
215             attackButtonArray[0][0].setStyle(
    buttonArray[row-1][column-1].getStyle());
216         }
217         else {
218             attackButtonArray[0][0].setStyle("-fx-
    border-color: Black; -fx-background-color: Black;");
219             attackButtonArray[0][0].setDisable(true
```

```
219 );
220         }
221         attackButtonArray[1][1].setStyle("-fx--fx-
    border-color: Navy Blue; -fx-background-color: Navy
    Blue;");
222     }
223 
224     public void setUpAttackGridPane() {
225         for(int row =0;row<attackButtonArray.length;
    row++) {
226             for(int column=0;column<
    attackButtonArray[0].length;column++) {
227                 attackButtonArray[row][column] = new
     Button("");
228                 attackButtonArray[row][column].
    setPrefHeight(200);
229                 attackButtonArray[row][column].
    setPrefWidth(200);
230                 attackButtonArray[row][column].
    setStyle("-fx-background-radius: 0px; -fx-border-
    color: LightBlue; -fx-background-color: LightBlue;"
    );
231                 AttackGridPane.add(attackButtonArray
    [row][column],column,row);
232             }
233         }
234         EventHandler x = new EventHandler<
    ActionEvent>() {
235             @Override
236             public void handle(ActionEvent event) {
237                 int row1 = GridPane.getRowIndex((
    Button) event.getSource());
238                 int column1 = GridPane.
    getColumnIndex((Button) event.getSource());
239                 Coordinates coords =
    findCoordinatesOnBoard(row1, column1);
240                 if (buttonClickedIsValid(row1,
    column1,coords)) {
241                     if (activatedButtons[row1][
    column1]) {
242                         attackButtonArray[row1][
```

```
242 column1].setStyle(allPlayers.get(turn%numPlayers).
    getColor() +" -fx-border-color: Black; -fx-border-
    width: 0px;");
243                         activatedButtons[row1][
    column1] = false;
244                     }
245                     else {
246                         activatedButtons[row1][
    column1] = true;
247                         attackButtonArray[row1][
    column1].setStyle(allPlayers.get(turn%numPlayers).
    getColor()+" -fx-border-color: #ff00ab; -fx-border-
    width: 5px;");
248                     }
249                     tallyUpTotals();
250                 }
251             }
252         };
253         for(int row =0;row<attackButtonArray.length;
    row++) {
254             for(int column=0;column<
    attackButtonArray[0].length;column++) {
255                 attackButtonArray[row][column].
    setOnAction(x);
256             }
257         }
258     }
259     private double winProbability = .50;
260     public void tallyUpTotals() {
261         double yourTotal = 0;
262         int enemyTotal = territoryArray[row][column
    ].getAmountOfSoldiers();
263         for(int row =0;row<3;row++) {
264             for(int column=0;column<3;column++) {
265                 if (activatedButtons[row][column]) {
266                     Coordinates coords =
    findCoordinatesOnBoard(row,column);
267                     yourTotal = yourTotal +
    territoryArray[coords.getRow()][coords.getColumn()].
    getAmountOfSoldiers();
268                 }
```

```
269             }
270         }
271         winProbability = .50 + ((yourTotal-
    enemyTotal)/10);
272         if (winProbability>.99) {
273             winProbability = 1;
274         }
275         if (winProbability<0) {
276             winProbability = 0;
277         }
278         DecimalFormat df = new DecimalFormat("#.##"
    );
279         operationBoxLabel1.setText("Your Soldiers: "
    +df.format(yourTotal));
280         operationBoxLabel2.setText("Enemy Soldiers
    : "+enemyTotal);
281         operationBoxLabel3.setText("Win Percentage
    : "+df.format(winProbability*100)+"%");
282     }
283 
284     public boolean buttonClickedIsValid(int row2,
    int column2,Coordinates coords) {
285         return territoryArray[coords.getRow()][
    coords.getColumn()].getControllingPlayer().getColor
    ().equals(allPlayers.get(turn%numPlayers).getColor
    ());
286     }
287 
288     public Coordinates findCoordinatesOnBoard(int
    row3, int column3) {
289         if (row3==0 && column3==0) {
290             return new Coordinates(row-1,column-1);
291         }
292         if (row3==0 && column3==1) {
293             return new Coordinates(row-1,column);
294         }
295         if (row3==0 && column3==2) {
296             return new Coordinates(row-1,column+1);
297         }
298         if (row3==1 && column3==0) {
299             return new Coordinates(row,column-1);
```

```
300         }
301         if (row3==1 && column3==1) {
302             return new Coordinates(row,column);
303         }
304         if (row3==1 && column3==2) {
305             return new Coordinates(row,column+1);
306         }
307         if (row3==2 && column3==0) {
308             return new Coordinates(row+1,column-1);
309         }
310         if (row3==2 && column3==1) {
311             return new Coordinates(row+1,column);
312         }
313         if (row3==2 && column3==2) {
314             return new Coordinates(row+1,column+1);
315         }
316         return new Coordinates(0,0);
317     }
318 
319     public void openAttackPlan() {
320         setUpSecondary();
321         boolean printOnLeft = getPrintSide();
322         attackBoxUI(true,false,printOnLeft);
323     }
324 
325     public boolean getPrintSide() {
326         return column>6;
327     }
328 
329     public void attackBoxUI(boolean visisble,
    boolean disable,boolean leftSide) {
330         blurRectangle.setVisible(visisble);
331         blurRectangle.setDisable(disable);
332         operationBOX.setVisible(visisble);
333         operationBOX.setDisable(disable);
334         if (visisble) {
335             blurRectangle.toFront();
336             operationBOX.toFront();
337         }
338         else {
339             blurRectangle.toBack();
```

```
340             operationBOX.toBack();
341             secondaryGridPane.toBack();
342         }
343         if (leftSide) {
344             operationBOX.setLayoutX(25);
345         }
346         else {
347             operationBOX.setLayoutX(775);
348         }
349 
350     }
351 
352     public void setUpSecondary() {
353         secondaryGridPane.toFront();
354         for(int row =0;row<displayButtonArray.length
    ;row++) {
355             for(int column=0;column<
    displayButtonArray[0].length;column++) {
356                 displayButtonArray[row][column].
    setStyle("-fx-border-color: transparent; -fx-
    background-color: transparent;");
357             }
358         }
359         displayButtonArray[row][column].setStyle(
    buttonArray[row][column].getStyle());
360     }
361 
362     public boolean validTerritory(int rowClicked,
    int columnClicked, boolean editing) {
363         if (editing) {
364             return territoryArray[rowClicked][
    columnClicked].getControllingPlayer().getColor().
    equals(allPlayers.get(turn%numPlayers).getColor());
365         }
366         else {
367             for (int i =0;i<allPlayers.get(turn%
    numPlayers).getAttackTerritories().size();i++) {
368                 if (allPlayers.get(turn%numPlayers).
    getAttackTerritories().get(i).getCoordOfTerritory().
    getRow()==rowClicked && allPlayers.get(turn%
    numPlayers).getAttackTerritories().get(i).
```

```
368 getCoordOfTerritory().getColumn()==columnClicked) {
369                     return true;
370                 }
371             }
372         }
373         return false;
374     }
375 
376     public void playTurn() {
377         setUpEditingMode();
378     }
379 
380     public void setUpEditingMode() {
381         MainDisplayLabel.setText("Editing Mode.
    Click on your territories to upgrade them.");
382         currentTurnLabel.setText("Current Turn:");
383         turnDisplayButton.setDisable(false);
384         turnDisplayButton.setVisible(true);
385         editingMode = true;
386         turnDisplayButton.setStyle("-fx-background-
    radius: 0px;");
387         turnDisplayButton.setStyle(allPlayers.get(
    turn%numPlayers).getColor());
388     }
389 
390     public void onClickEndEditingMode() {
391         setUpAttackingMode();
392         generateAreasToAttack();
393     }
394     public void generateAreasToAttack() {
395         allPlayers.get(turn%numPlayers).
    getAttackTerritories().clear();
396         for (int i = 0; i<allPlayers.get(turn%
    numPlayers).getAllTerritoriesControlled().size();i
    ++) {
397             ArrayList<Boolean> directionsCanGo =
    findDirections(allPlayers.get(turn%numPlayers).
    getAllTerritoriesControlled().get(i));
398             convertDirectionsToCoordinates(
    directionsCanGo, allPlayers.get(turn%numPlayers).
    getAllTerritoriesControlled().get(i));
```

```
399         }
400         removeRedundantTerritories();
401     }
402 
403     public void removeRedundantTerritories() {
404         for (int i = 0; i<allPlayers.get(turn%
    numPlayers).getAttackTerritories().size();i++) {
405             if(allPlayers.get(turn%numPlayers).
    getAttackTerritories().get(i).getControllingPlayer
    ().getColor().equals(allPlayers.get(turn%numPlayers
    ).getColor())) {
406                 allPlayers.get(turn%numPlayers).
    getAttackTerritories().remove(i);
407                 i--;
408             }
409         }
410     }
411 
412     public void convertDirectionsToCoordinates(
    ArrayList<Boolean> directions, Territory
    territoryChecking) {
413         if (directions.get(0)) {
414             allPlayers.get(turn%numPlayers).
    getAttackTerritories().add(territoryArray[
    territoryChecking.getCoordOfTerritory().getRow()-1][
    territoryChecking.getCoordOfTerritory().getColumn
    ()]);
415         }
416         if (directions.get(1)) {
417             allPlayers.get(turn%numPlayers).
    getAttackTerritories().add(territoryArray[
    territoryChecking.getCoordOfTerritory().getRow()-1][
    territoryChecking.getCoordOfTerritory().getColumn()+
    1]);
418         }
419         if (directions.get(2)) {
420             allPlayers.get(turn%numPlayers).
    getAttackTerritories().add(territoryArray[
    territoryChecking.getCoordOfTerritory().getRow()][
    territoryChecking.getCoordOfTerritory().getColumn()+
    1]);
```

```
421         }
422         if (directions.get(3)) {
423             allPlayers.get(turn%numPlayers).
    getAttackTerritories().add(territoryArray[
    territoryChecking.getCoordOfTerritory().getRow()+1][
    territoryChecking.getCoordOfTerritory().getColumn()+
    1]);
424         }
425         if (directions.get(4)) {
426             allPlayers.get(turn%numPlayers).
    getAttackTerritories().add(territoryArray[
    territoryChecking.getCoordOfTerritory().getRow()+1][
    territoryChecking.getCoordOfTerritory().getColumn
    ()]);
427         }
428         if (directions.get(5)) {
429             allPlayers.get(turn%numPlayers).
    getAttackTerritories().add(territoryArray[
    territoryChecking.getCoordOfTerritory().getRow()+1][
    territoryChecking.getCoordOfTerritory().getColumn()-
    1]);
430         }
431         if (directions.get(6)) {
432             allPlayers.get(turn%numPlayers).
    getAttackTerritories().add(territoryArray[
    territoryChecking.getCoordOfTerritory().getRow()][
    territoryChecking.getCoordOfTerritory().getColumn()-
    1]);
433         }
434         if (directions.get(7)) {
435             allPlayers.get(turn%numPlayers).
    getAttackTerritories().add(territoryArray[
    territoryChecking.getCoordOfTerritory().getRow()-1][
    territoryChecking.getCoordOfTerritory().getColumn()-
    1]);
436         }
437     }
438
439     public ArrayList<Boolean> findDirections(
    Territory territoryChecking) {
440         ArrayList<Boolean> tempList = new ArrayList
```

```
440 <>();
441         for (int i = 0 ;i<8;i++) {
442             tempList.add(false);
443         }
444         if(territoryChecking.getCoordOfTerritory().
    getColumn()+1<=(buttonArray[0].length-1)) {
445             tempList.set(2,true);
446         }
447         if(territoryChecking.getCoordOfTerritory().
    getColumn()>0) {
448             tempList.set(6,true);
449         }
450         if(territoryChecking.getCoordOfTerritory().
    getRow()+1<=(buttonArray.length-1)) {
451             tempList.set(4,true);
452         }
453         if(territoryChecking.getCoordOfTerritory().
    getRow()>0) {
454             tempList.set(0,true);
455         }
456         if (tempList.get(0) && tempList.get(2)) {
457             tempList.set(1,true);
458         }
459         if (tempList.get(2) && tempList.get(4)) {
460             tempList.set(3,true);
461         }
462         if (tempList.get(4) && tempList.get(6)) {
463             tempList.set(5,true);
464         }
465         if (tempList.get(0) && tempList.get(6)) {
466             tempList.set(7,true);
467         }
468         return tempList;
469     }
470
471     public void setUpAttackingMode() {
472         editingMode = false;
473         attacksLeft = 3;
474         MainDisplayLabel.setText("Attack Mode. Click
     On The Surrounding Territories To Invade.");
475         currentTurnLabel.setText("Current Turn:");
```

```
476         turnDisplayButton.setDisable(false);
477         turnDisplayButton.setVisible(true);
478         turnDisplayButton.setStyle("-fx-background-
    radius: 0px;");
479         turnDisplayButton.setStyle(allPlayers.get(
    turn%numPlayers).getColor());
480         attacksLeftLabel.setText("Attacks Left: "+
    attacksLeft);
481     }
482 
483 
484     public void setUpPlayers() {
485         for (int i =0;i<numPlayers;i++) {
486             if (i==0) {
487                 allPlayers.add(new Player("temp",1,
    new Territory(0,0,10,1), "-fx-background-color: Blue
    ;"));
488                 territoryArray[0][0].
    setControllingPlayer(allPlayers.get(allPlayers.size
    ()-1));
489             }
490             else if (i==1) {
491                 allPlayers.add(new Player("temp",1,
    new Territory(0,14,10,1), "-fx-background-color: Red
    ;"));
492                 territoryArray[0][14].
    setControllingPlayer(allPlayers.get(allPlayers.size
    ()-1));
493             }
494             else if (i==2) {
495                 allPlayers.add(new Player("temp",1,
    new Territory(9,0,10,1), "-fx-background-color:
    Yellow;"));
496                 territoryArray[9][0].
    setControllingPlayer(allPlayers.get(allPlayers.size
    ()-1));
497             }
498             else {
499                 allPlayers.add(new Player("temp",1,
    new Territory(9,14,10,1), "-fx-background-color:
    Green;"));
```

```
500                 territoryArray[9][14].
    setControllingPlayer(allPlayers.get(allPlayers.size
    ()-1));
501             }
502         }
503         updateMap();
504     }
505 
506     public void updateMap() {
507         setUpImageViews();
508         for(int row =0;row<territoryArray.length;row
    ++) {
509             for(int column=0;column<territoryArray[0
    ].length;column++) {
510                 buttonArray[row][column].setStyle(
    territoryArray[row][column].getControllingPlayer().
    getColor());
511                 buttonArray[row][column].setGraphic(
    imageViewArray[row][column]);
512             }
513         }
514     }
515 
516     public void setUpImageViews() {
517         for(int row =0;row<imageViewArray.length;row
    ++) {
518             for(int column=0;column<imageViewArray[0
    ].length;column++) {
519                 try {
520                     if (territoryArray[row][column].
    getTerritoryTypeID()==0) {
521 
522                     }
523                     else if (territoryArray[row][
    column].getTerritoryTypeID()==1) {
524                         FileInputStream input = new
    FileInputStream("src/main/resources/images/homeBase.
    png");
525                         imageViewArray[row][column].
    setImage(new Image(input));
526                     }
```

```
527                 }catch (FileNotFoundException e) {
528                     e.printStackTrace();
529                 }
530             }
531         }
532     }
533 
534     public void askForAmountOfPlayers() {
535         boolean enteredProperly = false;
536         while (!enteredProperly) {
537             numPlayers = Integer.parseInt(
    JOptionPane.showInputDialog("how many people will be
     playing (2-4)"));
538             if (numPlayers>1 && numPlayers<5) {
539                 enteredProperly = true;
540             }
541         }
542     }
543 
544     public void printWorldBorders() {
545         try{
546             FileReader reader = new FileReader("src/
    main/resources/tempCoordinates.txt");
547             Scanner in = new Scanner(reader);
548             ArrayList<String> allCords = new
    ArrayList<>();
549             while (in.hasNext()) {
550                 allCords.add(in.nextLine());
551             }
552             for (int i =0;i<allCords.size();i++) {
553                 int spot = 0;
554                 StringBuilder num = new
    StringBuilder();
555                 while (allCords.get(i).charAt(spot
    ) != ',') {
556                     num.append(allCords.get(i).
    charAt(spot));
557                     spot++;
558                 }
559                 StringBuilder num1 = new
    StringBuilder();
```

```
560                 for (int j = spot+1;j<allCords.get(i
    ).length();j++) {
561                     num1.append(allCords.get(i).
    charAt(j));
562                 }
563                 buttonArray[Integer.parseInt(String.
    valueOf(num))][Integer.parseInt(String.valueOf(num1
    ))].setStyle("-fx-border-color: Blue; -fx-background
    -color: Blue;");
564             }
565         }catch (FileNotFoundException variable){
566             System.out.println("file not found");
567         }
568     }
569
570     public void addCoordinatesToTXT(int rowOfClick,
    int columnOfClick) {
571         String outFile = "src/main/resources/
    tempCoordinates.txt";
572         try{
573             FileReader reader = new FileReader("src/
    main/resources/tempCoordinates.txt");
574             Scanner in = new Scanner(reader);
575             ArrayList<String> allCords = new
    ArrayList<>();
576             while (in.hasNext()) {
577                 allCords.add(in.nextLine());
578             }
579             PrintWriter out = new PrintWriter(
    outFile);
580             for (int i = 0;i<allCords.size();i++) {
581                 out.println(allCords.get(i));
582             }
583             out.println(rowOfClick +","+
    columnOfClick);
584             out.close();
585         }catch (FileNotFoundException variable){
586             System.out.println("file not found");
587         }
588     }
589     @FXML
```

```
590     public void onClickShowAttackOptions() {
591         if (!editingMode) {
592             if(possibleOptionsCheckBox.isSelected
    ()) {
593                 for (int i =0;i<allPlayers.get(turn%
    numPlayers).getAttackTerritories().size();i++) {
594                     buttonArray[allPlayers.get(turn%
    numPlayers).getAttackTerritories().get(i).
    getCoordOfTerritory().getRow()][allPlayers.get(turn%
    numPlayers).getAttackTerritories().get(i).
    getCoordOfTerritory().getColumn()].setStyle("-fx-
    border-color: Cyan; -fx-background-color: Cyan;");
595                 }
596             }
597             else {
598                 for (int i =0;i<allPlayers.get(turn%
    numPlayers).getAttackTerritories().size();i++) {
599                     buttonArray[allPlayers.get(turn%
    numPlayers).getAttackTerritories().get(i).
    getCoordOfTerritory().getRow()][allPlayers.get(turn%
    numPlayers).getAttackTerritories().get(i).
    getCoordOfTerritory().getColumn()].setStyle("-fx-
    border-color: LightBlue; -fx-background-color:
    LightBlue;");
600                 }
601             }
602         }
603     }
604
605     @FXML
606     public void cancelInvasion() {
607         attackBoxUI(false,true,true);
608     }
609
610     @FXML
611     public void attemptInvasion() {
612         double randInt = Math.random();
613         if (randInt<winProbability) {
614             territoryArray[row][column].
    setControllingPlayer(allPlayers.get(turn%numPlayers
    ));
```

```
615             allPlayers.get(turn%numPlayers).
    getAllTerritoriesControlled().add(territoryArray[row
    ][column]);
616             updateMap();
617             generateAreasToAttack();
618             attackBoxUI(false,true,true);
619             System.out.println("SUCCESS!!!");
620         }
621         else {
622             System.out.println("THE INVASION FAILED"
    );
623             attackBoxUI(false,true,true);
624         }
625         attacksLeft--;
626         attacksLeftLabel.setText("Attacks Left: "+
    attacksLeft);
627     }
628 }
```

```java
package com.example.template;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 1200, 900);
        stage.setTitle("Infrastructure Risk");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}
```