

CONCORDIA UNIVERSITY

**DEPARTMENT OF
COMPUTER SCIENCE AND SOFTWARE
ENGINEERING**

SOEN 6441, Winter 2019

Instructor: Amin Ranj Bar

RISK Game (Build-1)

Coding Standards

BY:

Team - 13

Jemish Kishor Paghadar (40080723)

Deep Alpesh Patel (40087798)

Raj Patel (40085012)

Hardik Vora (40087606)

Dolly Modha (40084358)

Coding Conventions

► Code Layout

- Blank lines have been left between code blocks to increase readability
- Curly braces are written on the same line on which code block starts, rather than on separate line.
- The body of a function indented with respect to its header.
- The body of a for, while, or switch statement is indented with respect to its first line; and similarly for if statements and other nested structures.



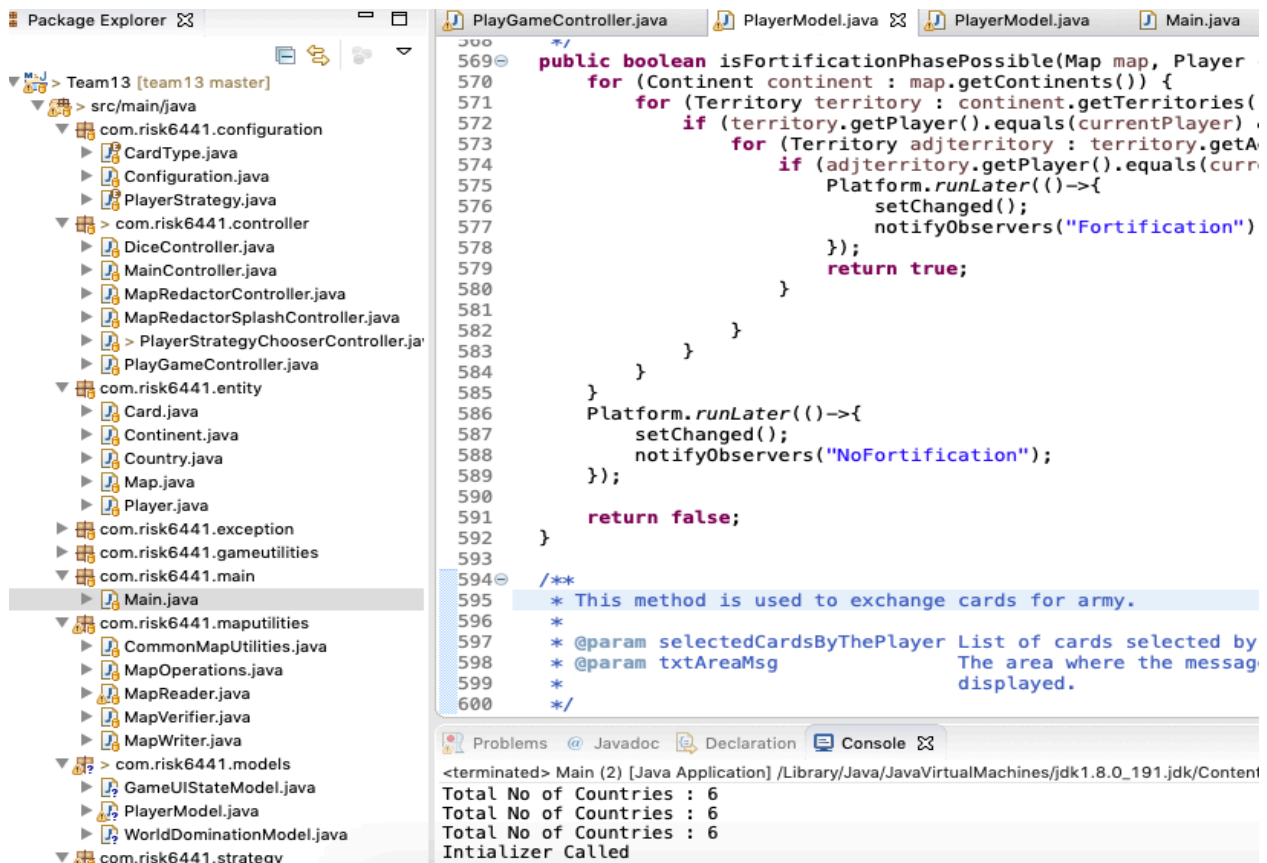
```
344 public void reinforcementPhase(Territory territory, ObservableList<Territory> terrList, TextArea txtAreaMsg) {
345     ArrayList<Territory> terrArList = new ArrayList<Territory>(terrList);
346
347     if(playerList.size() <= 1)
348         return;
349
350     // Run the task in a background thread
351     if(currentPlayer.getStrategy() instanceof Human || (!Config.isThreadingForTournament)) {
352         System.out.println("Inside this");
353         currentPlayer.getStrategy().reinforcementPhase(terrList, territory, currentPlayer, terrArList, null);
354         if (currentPlayer.getArmies() <= 0 && playerList.size() > 1) {
355             GameUtils.addToLog("===Reinforcement phase Ended! ===\n");
356             setChanged();
357             notifyObservers("Attack");
358         }
359     } else {
360         Thread backgroundThread = new Thread(new Runnable() {
361             @Override
362             public void run() {
363                 try {
364                     Thread.sleep(Config.waitBetweenTurn);
365                 } catch (InterruptedException e) {
366                     e.printStackTrace();
367                 }
368                 currentPlayer.getStrategy().reinforcementPhase(terrList, territory, currentPlayer, terrArList, null);
369                 if (currentPlayer.getArmies() <= 0 && playerList.size() > 1) {
370                     GameUtils.addToLog("===Reinforcement phase Ended! ===\n");
371                     Platform.runLater(() -> {
372                         setChanged();
373                         notifyObservers("updateReinforceArmy");
374                         setChanged();
375                         notifyObservers("Attack");
376                     });
377                 }
378             }
379         });
380         // Terminate the running thread if the application exits
381         backgroundThread.setDaemon(true);
382         // Start the thread
383         backgroundThread.start();
384     }
385 }
386 }
```

► Indentation

- Code is indented according to its nesting level to improve code readability.

► File Organization

- The package name should be in lowercase. For this project, every package name starts with com.risk6441 and then append with the functionality that the classes in package are going to implement. ex. **com.risk6441.models**.
- Any external file that is being used such as map file, should be in lowercase and should be in resource folder.



► Naming Conventions

- Constant have been named with uppercase letters and separated by underscore.

```

24     public static final Integer ARMIES_THREE_PLAYER = 35;
25
26     /**
27      * The ARMIES_TWO_PLAYER Constant assigns 30 armies to 4 player in the game
28      */
29     public static final Integer ARMIES_FOUR_PLAYER = 30;

```

- All class names start with uppercase letter and words are separated using case change.
- Folder and Package names are written in lowercase.
- File names are written in UpperCamelCase.

```

> src/main/java
  > com.risk6441.configuration
    > CardType.java
      > CardType

```

- Method and Parameter names are written in lowerCamelCase.
- Local variable names are written in lowerCamelCase.

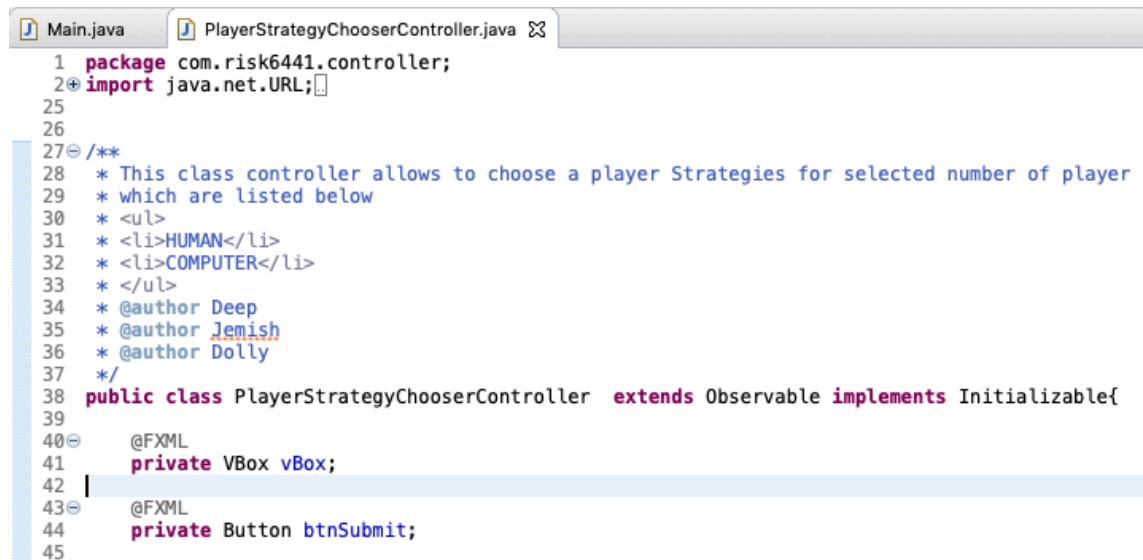
```

26     private Player player;
27     private List<Country> adjacentCountries;
28     private boolean isProcessed;
29
30     /**
31      * Setter method for the player.
32      * @param player set player
33      */
34     public void setPlayer(Player player) {
35         this.player = player;
36     }

```

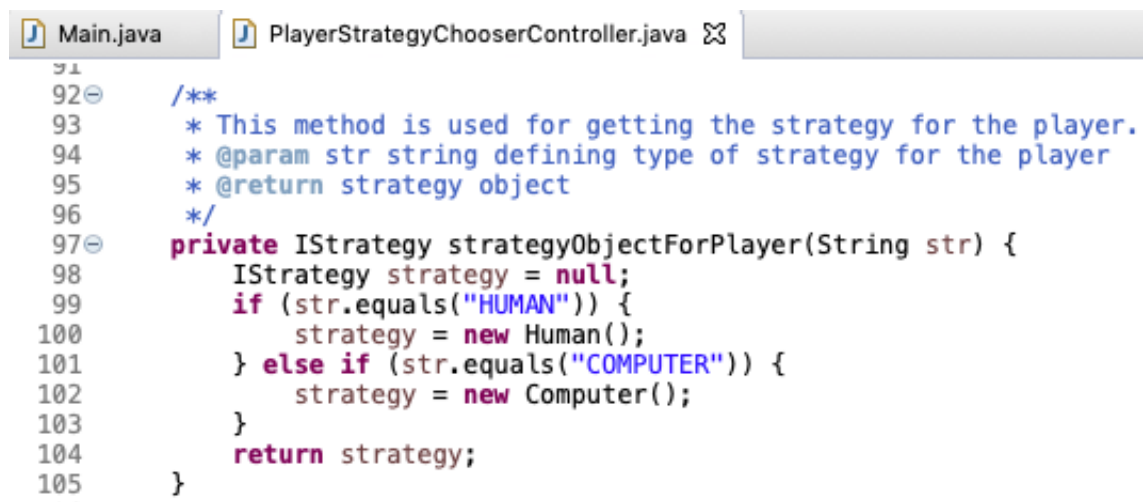
► Comments

- Commenting is done as per conventions for Java Doc.
- Each class declaration precedes by a comment explaining what the class is for.



```
1 package com.risk6441.controller;
2 import java.net.URL;
25
26
27 /**
28  * This class controller allows to choose a player Strategies for selected number of player
29  * which are listed below
30  * <ul>
31  * <li>HUMAN</li>
32  * <li>COMPUTER</li>
33  * </ul>
34  * @author Deep
35  * @author Jemish
36  * @author Dolly
37  */
38 public class PlayerStrategyChooserController extends Observable implements Initializable{
39
40     @FXML
41     private VBox vbox;
42
43     @FXML
44     private Button btnSubmit;
45
```

- Each method or function have comments explaining what it does, as well as what is the purpose of parameters and return type description if the method's return is non-void.



```
92 /**
93  * This method is used for getting the strategy for the player.
94  * @param str string defining type of strategy for the player
95  * @return strategy object
96  */
97 private IStrategy strategyObjectForPlayer(String str) {
98     IStrategy strategy = null;
99     if (str.equals("HUMAN")) {
100         strategy = new Human();
101     } else if (str.equals("COMPUTER")) {
102         strategy = new Computer();
103     }
104     return strategy;
105 }
```