CONCORDIA UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

SOEN 6441, Winter 2019

Instructor: Amin Ranj Bar

RISK Game (Build-3)

Coding Standards

BY:

Team - 13

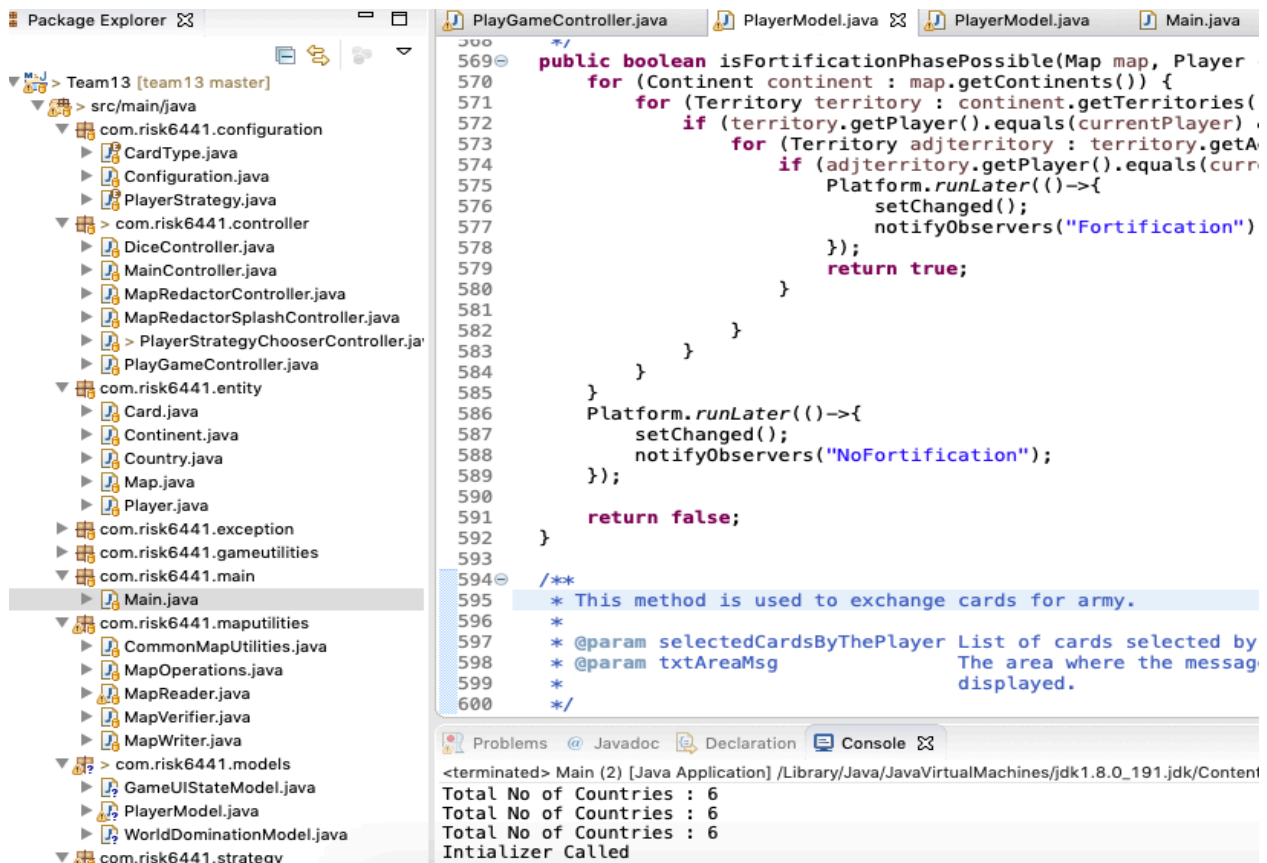|	
Jemish Kishor Paghadar	(40080723)
Deep Alpesh Patel	(40087798)
Raj Patel	(40085012)
Hardik Vora	(40087606)
Dolly Modha	(40084358)

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.

```java
public void reinforcementPhase(Territory territory, ObservableList<Territory> terrList, TextArea txtAreaMsg) {
    ArrayList<Territory> terrArList = new ArrayList<Territory>(terrList);

    if(playerList.size()<=1)
        return;

    // Run the task in a background thread
    if(currentPlayer.getStrategy() instanceof Human || (!Config.isThreadingForTournament)) {
        System.out.println("Inside this");
        currentPlayer.getStrategy().reinforcementPhase(terrList, territory, currentPlayer,terrArList, null);
        if (currentPlayer.getArmies() <= 0 && playerList.size() > 1) {
            GameUtils.addTextToLog("===Reinforcement phase Ended! ===\n");
            setChanged();
            notifyObservers("Attack");
        }
    }else {
        Thread backgroundThread = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep(Config.waitBeweenTurn);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                currentPlayer.getStrategy().reinforcementPhase(terrList, territory, currentPlayer,terrArList,null);
                if (currentPlayer.getArmies() <= 0 && playerList.size() > 1) {
                    GameUtils.addTextToLog("===Reinforcement phase Ended! ===\n");
                    Platform.runLater(() -> {
                        setChanged();
                        notifyObservers("updateReinforceArmy");
                        setChanged();
                        notifyObservers("Attack");
                    });
                }
            }
        });
        // Terminate the running thread if the application exits
        backgroundThread.setDaemon(true);
        // Start the thread
        backgroundThread.start();
    }
}
```

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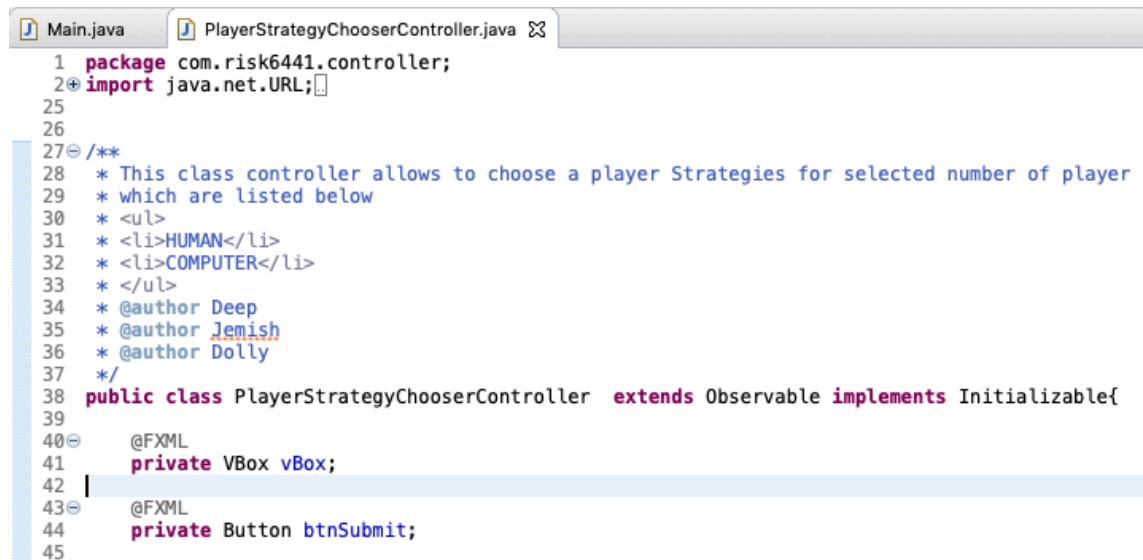
```
26        private Player player;
27        private List<Country> adjacentCountries;
28        private boolean isProcessed;
29
30⊝        /**
31         * Setter method for the player.
32         * @param player set player
33         */
34⊝        public void setPlayer(Player player) {
35        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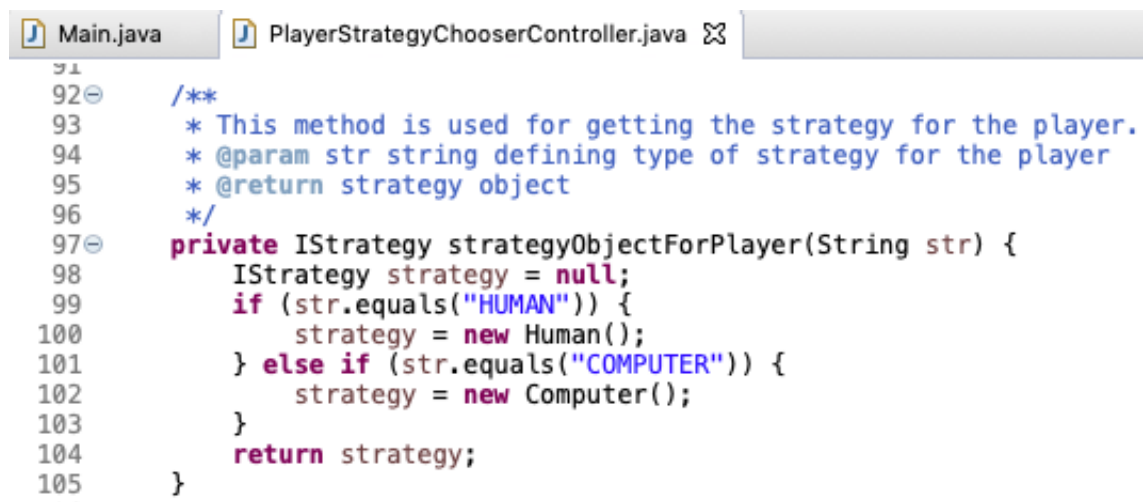
‣ Comments

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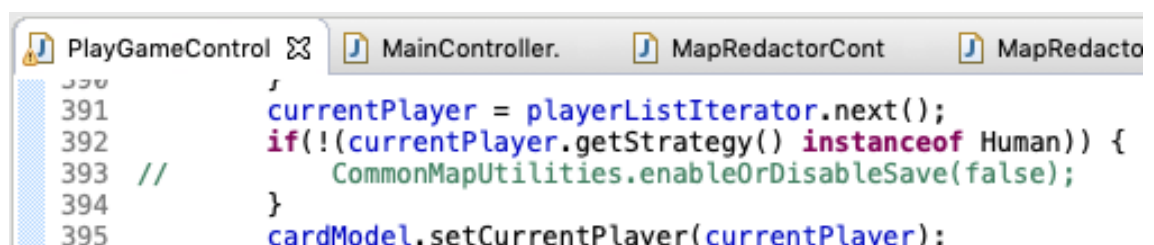- Each class declaration precedes by a comment explaining what the class is for.

```java
  J Main.java    J PlayerStrategyChooserController.java
  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.
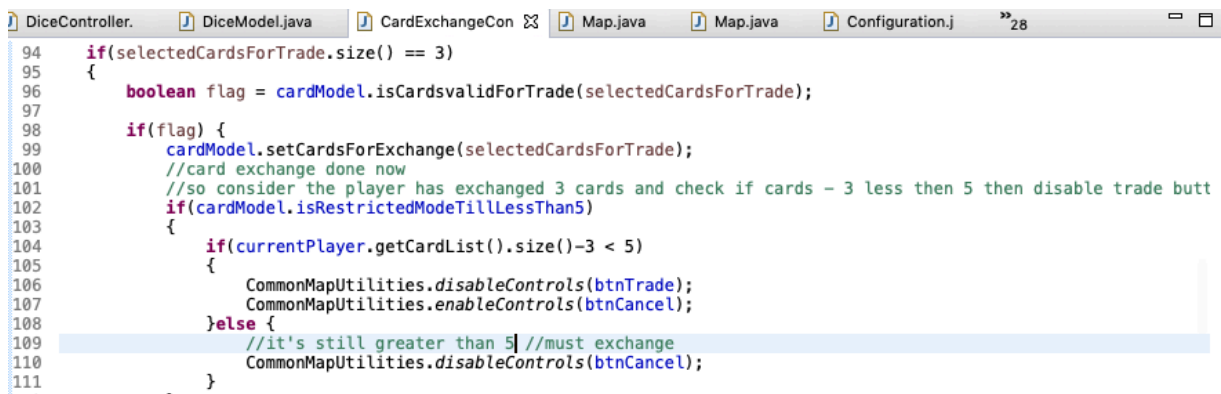
```java
  J Main.java    J PlayerStrategyChooserController.java
 91
 92⊝     /**
 93       * This method is used for getting the strategy for the player.
 94       * @param str string defining type of strategy for the player
 95       * @return strategy object
 96       */
 97⊝     private IStrategy 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       }
```

- "commented out" code

```java
  J PlayGameControl    J MainController.    J MapRedactorCont    J MapRedacto
 390    J
 391             currentPlayer = playerListIterator.next();
 392             if(!(currentPlayer.getStrategy() instanceof Human)) {
 393  //              CommonMapUtilities.enableOrDisableSave(false);
 394             }
 395             cardModel.setCurrentPlayer(currentPlayer);
```

- Long inline comments for code description

```
94  if(selectedCardsForTrade.size() == 3)
95  {
96      boolean flag = cardModel.isCardsvalidForTrade(selectedCardsForTrade);
97
98      if(flag) {
99          cardModel.setCardsForExchange(selectedCardsForTrade);
100         //card exchange done now
101         //so consider the player has exchanged 3 cards and check if cards – 3 less then 5 then disable trade butt
102         if(cardModel.isRestrictedModeTillLessThan5)
103         {
104             if(currentPlayer.getCardList().size()-3 < 5)
105             {
106                 CommonMapUtilities.disableControls(btnTrade);
107                 CommonMapUtilities.enableControls(btnCancel);
108             }else {
109                 //it's still greater than 5 //must exchange
110                 CommonMapUtilities.disableControls(btnCancel);
111             }
```
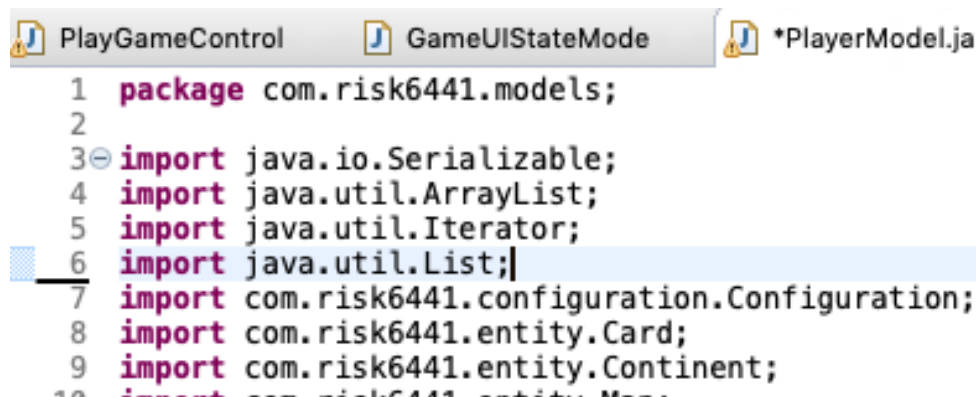
‣ **Indentation**

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

‣ **Packages** [1]

- The prefix of a unique package name is always written in **all-lowercase ASCII letters** and should be one of the top-level domain names, like com, edu, gov, mil, net, org.
- Subsequent components of the package name vary according to an organisation's own internal naming conventions.

```
1  package com.risk6441.models;
2
3  import java.io.Serializable;
4  import java.util.ArrayList;
5  import java.util.Iterator;
6  import java.util.List;
7  import com.risk6441.configuration.Configuration;
8  import com.risk6441.entity.Card;
9  import com.risk6441.entity.Continent;
```
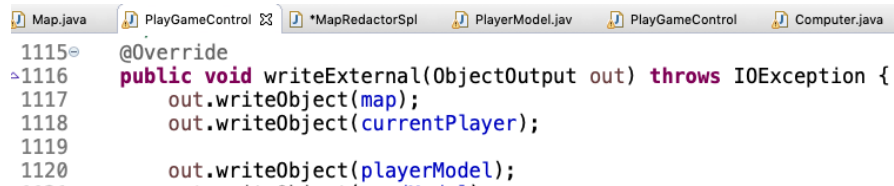
‣ **Caught exceptions**[2]

- In program it is appropriate to take no action whatsoever in a catch block

```
Parent root = null;
try {
    root = (Parent) loader.load();
} catch (IOException e) {

}
Stage stage = new Stage();
Scene scene = new Scene(root);
```

- Use throws to handle method level or class level exception

- Whenever you specify an exception in your method signature, you should also document it in your Javadoc.

References :

1. https://www.geeksforgeeks.org/java-naming-conventions/
2. https://google.github.io/styleguide/javaguide.html#s6.2-caught-exceptions