

## **TEAM 14 : REFACTORING**

**COURSE : SOEN 6441 - WINTER 2024**

**INSTRUCTOR : JOEY PAQUET**

### **TEAM MEMBERS:**

1. Sanjay Bhargav Pabbu
2. Blesslin Jeba Shiny Augustin Moses
3. Susmitha Mamula
4. Poojitha Bhupali
5. Piyush Gupta
6. Mahfuzzur Rahman

### **POTENTIAL REFACTORING TARGETS:**

1. The State Pattern has been successfully executed.
2. The Observer Pattern has been implemented.
3. The Command Pattern has been implemented.
4. Dead code has been removed.
5. The logic of setting the continent list has been moved from the startup phase to the reinforcement assignment phase.
6. A new 'owner' field has been introduced in the Country Class, with respective getter and setter methods created.
7. Improvements have been made to the UI in the 'showmap' functionality and 'showCommands()' method.
8. The signature of the 'IssueOrder()' method has been changed.
9. The Order Class has been updated to an interface.
10. Redundant code in the GameEngine has been removed.
11. The implementation of 'deploy' from the Order Class (now interface) has been moved to the Deploy Class.
12. A separate method 'isValid()' has been created in the Player Controller Class for command validation within the 'issueOrder()' loop.
13. A separate algorithm has been implemented to prompt the player to re-enter the order command if it is syntactically invalid.
14. Additional fields have been added in the Player Class to support new functionalities.
15. A new logic has been implemented to end the game when a player wins.

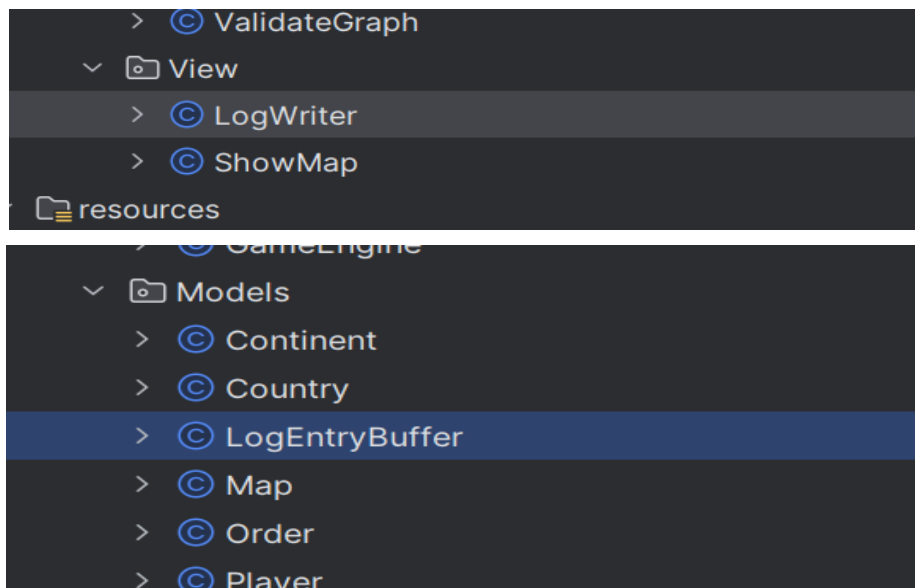
### **ACTUAL REFACTORING TARGETS:**

#### **Before/After Refactoring**

## 1. Implementing Observer Pattern for Console Log

### Refactoring

- As a part of refactoring mentioned in build 2, the observer pattern is implemented which writes to the console using the LogWriter. The refactoring process got it to writing in the logfile in the console which also enhanced the application overall. For each action performed throughout the game, such as executing a command or issuing/fulfilling an order, a LogEntryBuffer object is populated with details describing the outcome of the action.



```
1  package org.concordia.macs.View;
2  import org.concordia.macs.Utilities.Observer;
3
4  import java.io.BufferedWriter;
5  import java.io.File;
6  import java.io.FileWriter;
7  import java.io.PrintWriter;
8
9  /**
10   * This class writes the log into the log file.
11   * @author - Blesslin Jeba Shiny
12   */
13  public class LogWriter implements Observer {
14      private String l_fileName="LogEntry";
15
16      /**
17       * Observer interface implementation
18       * @param p_log - log message to be updated
19       */
20      @Override
21      public void update(String p_log)
22      {
23          writeLog(p_log);
24      }
```

**LogWriter implements Observer, to write in the log file**

2. Extend the State Pattern to cover Startup, Issue, and Order phases for better organization and adaptability.

### Refactoring

- The refactoring was done as a part of implementing and extending the state pattern to cover the startup, issue and order phases to ensure proper functioning and enhanceability.

### Code

```
1  package org.concordia.macs.State;
2  import java.util.ArrayList;
3  import org.concordia.macs.Controllers.GameEngine;
4  import org.concordia.macs.Models.Continent;
5  import org.concordia.macs.Models.Country;
6  import org.concordia.macs.Models.LogEntryBuffer;
7  import org.concordia.macs.Models.Player;
8  import org.concordia.macs.Utilities.Connectivity;
9  import org.concordia.macs.View.ShowMap;
10 /**
11  * @author Susmitha Mamula
12  * This class manages the different states within the State Pattern in the game.
13  */
14
15  public abstract class Phase {
16
17
18
19      GameEngine ge;
20
21      Phase(GameEngine p_g)
22      {
23          ge = p_g;
24      }
```

3. PlayersGameplay being modified with attack, bomb, AirliftDeploy, Blockade

### Refactoring

- Refactoring done as a part to upgrade the gameplay from build 1 by including these methods and functionalities

## Code

```
1 Piyush Gupta +1
public static int attack(Player p_player, ArrayList<Player> p_playersArray, Country p_fromCountry,
                        Country p_toCountry, int p_troops, ArrayList<Continent> p_continent, Connectivity p_connectivity) {
    if(p_troops<0) {
        System.out.println(ColorCoding.ANSI_RED+"Error: Can't attack with negative number of troops"+ColorCoding.ANSI_RESET);
        return 1;
    }

    try {
        if(!p_connectivity.getD_countriesList().contains(p_fromCountry) || !p_connectivity.getD_countriesList().contains(p_toCountry)) {
            if(!p_connectivity.getD_countriesList().contains(p_fromCountry)) {
                System.out.println(ColorCoding.ANSI_RED+"Error: Country "+p_fromCountry.getD_countryName()+" not found"+ColorCoding.ANSI_RESET);
            }
            if(!p_connectivity.getD_countriesList().contains(p_toCountry)) {
                System.out.println(ColorCoding.ANSI_RED+"Error: Country "+p_toCountry.getD_countryName()+" not found"+ColorCoding.ANSI_RESET);
            }
            return 1;
        }
    } catch (Exception e) {
        System.out.println(ColorCoding.ANSI_RED+"Error: "+e.getMessage()+ColorCoding.ANSI_RESET);
        return 1;
    }
}
```

```
public static int bomb(Player p_player, ArrayList<Player> p_playersArray, Country p_toCountry, ArrayList<Continent> p_continent, Connectivity p_connectivity) {
    LogEntryBuffer d_logEntryBuffer = new LogEntryBuffer();
    int l_targetArmies = 0;
    int l_countryFoundFlag = 0;
    for (Country c : p_player.getD_country()) {
        if (c.getD_neighbours().contains(p_toCountry.getD_countryId())) {
            l_countryFoundFlag++;
        }
    }
    if (l_countryFoundFlag != 0) {
        // Bomb logic
    }
}
```

```
public static boolean AirliftDeploy(String p_sourceCountryObj, String p_targetCountryObj, int p_armiesToAirlift, Player p_player, ArrayList<Player> p_playersArray, ArrayList<Continent> p_continent, Connectivity p_connectivity) {
    LogEntryBuffer d_logEntryBuffer = new LogEntryBuffer();

    String l_sourceCountryVar = p_sourceCountryObj;
    String l_targetCountryVar = p_targetCountryObj;

    int l_armiesToAirlift = p_armiesToAirlift;

    Player l_player = p_player;
}
```

```
1 usage 1 Piyush Gupta +1
@ public static boolean Blockade(String p_sourceCountryObj, Player p_player, ArrayList<Player> p_playersArray, ArrayList<Continent> p_continent, Connectivity p_connectivity) {
    LogEntryBuffer d_logEntryBuffer = new LogEntryBuffer();

    String l_sourceCountryVar = p_sourceCountryObj;

    Player l_player = p_player;

    int flag = 0;

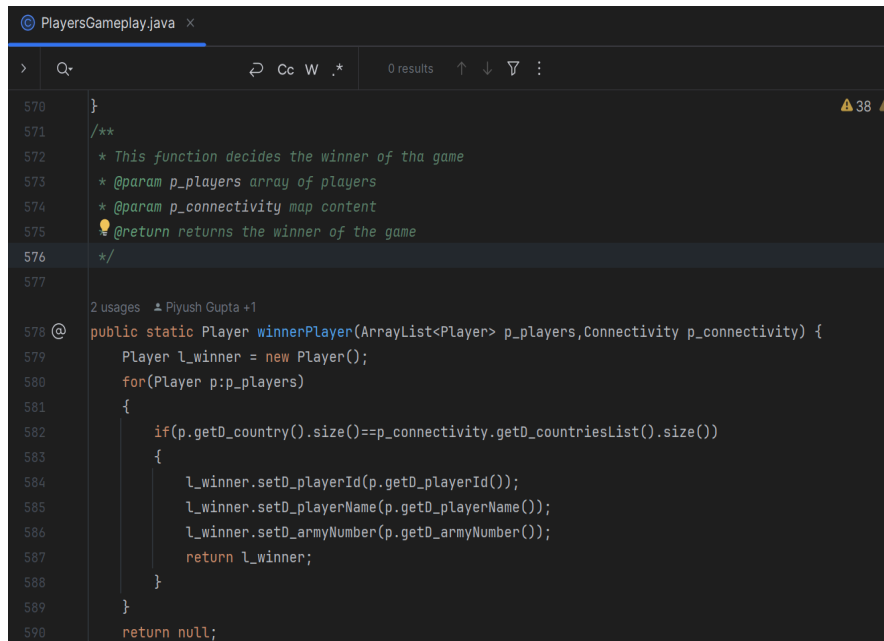
    ArrayList<Country> l_country = new ArrayList<>();
    l_country = (ArrayList<Country>) p_player.getD_country();
    Country l_sourceCountry = new Country();
}
```

## 4. Implemented new logic for game termination

- In the previous build, the game's end condition required all players to deplete their reinforcement pools and have no remaining armies. However, in the build 2 version, this condition has been revised to declare the player who conquers all countries on the map as the winner. To accommodate this adjustment, a new

method has been devised to validate this condition and smoothly transition the game into the "game over" phase.

## After Refactoring



```
570 }
571 /**
572  * This function decides the winner of the game
573  * @param p_players array of players
574  * @param p_connectivity map content
575  * @return returns the winner of the game
576  */
577
578 2 usages  Piyush Gupta +1
579 @ public static Player winnerPlayer(ArrayList<Player> p_players, Connectivity p_connectivity) {
580     Player l_winner = new Player();
581     for (Player p : p_players)
582     {
583         if (p.getD_country().size() == p_connectivity.getD_countriesList().size())
584         {
585             l_winner.setD_playerId(p.getD_playerId());
586             l_winner.setD_playerName(p.getD_playerName());
587             l_winner.setD_armyNumber(p.getD_armyNumber());
588             return l_winner;
589         }
590     }
591     return null;
592 }
```

## 5. Added new functionalities in 'Player' class

### Before Refactoring

- The class holds basic details of the player such as name, ID, and number of armies. It manages orders given by the player and the list of countries owned by the player.

```

C: > Users > Susmitha > Downloads > SOEN6441-Project > SOEN6441-Project > src > main > java > org > concordia
import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.List;
6
7  /**
8   * class with the details of the player
9   *
10  * @author Susmitha Mamula
11  */
12
13  public class Player {
14      private String d_playerName;
15      private int d_playerId;
16      private int d_armyNumber;
17
18      private Order d_order;
19      private List<Country> d_country = new ArrayList<>();
20      private List<Order> d_playerOrder = new ArrayList<>();
21      private List<Continent> d_continent = new ArrayList<>();
22
23      /**
24       * issues the orders given by the player
25       */
26      public void issue_order(){
27          this.d_playerOrder.add(d_order);
28      }
29
30      /**

```

### After Refactoring

- Card Management: The 'Player' class now includes functionality to manage cards owned by the player. Functions such as `getCards()`, `setCards()`, `removeCard()`, and `addCard()` to manipulate the list of cards owned by the player.
- Diplomacy Management: The 'Player' class now has functionality to manage diplomacy between players. It includes methods such as `getDiplomacyWith()`, `setDiplomacyWith()`, `addDiplomacyWith()`, and `clearDiplomacyWith()` to handle diplomacy relationships with other players.
- Additional methods like `removeAllCountryAndContinentAssigned()` and `removeCountry()` have been added to remove countries and continents assigned to the player.

```

205     /**
206      * This function returns the cards of the given player
207      * @return list of cards
208      */
209     public ArrayList<String> getCards() {
210         return (ArrayList<String>) d_cards;
211     }
212     /**
213      * This function sets the cards owned by the players
214      * @param cards cards owned by player
215      */
216
217     public void setCards(ArrayList<String> cards)
218     {
219         d_cards = cards;
220     }
221     /**
222      * This function removes the card from the list of cards owned by the player
223      * @param card Card to be removed from the array list
224      */
225     public void removeCard(String card)
226     {
227         d_cards.remove(card);
228     }
229     /**
230      * This function adds the card to the list of cards
231      * @param card refers to the card to be added to list
232      */
233     public void addCard(String card)
234     {
235         d_cards.add(card);
236     }

```

## Card Management Functionalities

```

237     /**
238      * This function returns the countries the player set the diplomacy with
239      * @return refers to the second country the player set the diplomacy with
240      */
241     public ArrayList<Integer> getDiplomacyWith()
242     {
243         return d_diplomacyWith;
244     }
245     /**
246      * This function sets the countries the player sets the diplomacy with
247      * @param diplomacyWith refers to the second country the player sets the diplomacy with
248      */
249     public void setDiplomacyWith(ArrayList<Integer> diplomacyWith)
250     {
251         d_diplomacyWith = diplomacyWith;
252     }
253     /**
254      * This function adds the country to the diplomacy list
255      * @param l_toPlayerID represents the playerId to perform diplomacy with
256      */
257     public void addDiplomacyWith(Integer l_toPlayerID)
258     {
259         d_diplomacyWith.add(l_toPlayerID);
260     }
261     /**
262      * This function clears the diplomacy list of the player
263      */
264     public void clearDiplomacyWith()
265     {
266         d_diplomacyWith.clear();
267     }

```

## Diplomacy Management Functionalities

Implemented java command line option -Xdoclint to prove that the Javadoc is complete.

```
[INFO] --- jar:3.3.0:jar (default-jar) @ SOEN6441-Project ---
[INFO] Building jar: C:\Users\pabbu\OneDrive\Documents\BuildTwo\SOEN6441-Project\target\SOEN6441-Project-1.0-SNAPSHOT.jar
[INFO] --- install:3.1.1:install (default-install) @ SOEN6441-Project ---
[INFO] Installing C:\Users\pabbu\OneDrive\Documents\BuildTwo\SOEN6441-Project\pom.xml to C:\Users\pabbu\.m2\repository\org\concordia\macs\SOEN6441-Project\1.0-SNAPSHOT\SOEN6441-Project-1.0-SNAPSHOT.pom
[INFO] Installing C:\Users\pabbu\OneDrive\Documents\BuildTwo\SOEN6441-Project\target\SOEN6441-Project-1.0-SNAPSHOT.jar to C:\Users\pabbu\.m2\repository\org\concordia\macs\SOEN6441-Project\1.0-SNAPSHOT\SOEN6441-Project-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.318 s
[INFO] Finished at: 2024-03-17T23:06:56-04:00
[INFO] -----
```

Dependency added in pom.xml for Xdoclint

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-javadoc-plugin</artifactId>
  <version>3.1.1</version>
  <configuration>
    <doclint>all</doclint>
  </configuration>
  <executions>
    <execution>
      <id>attach-javadocs</id>
      <goals>
        <goal>jar</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```