

# REFACTORING DOCUMENT

## 1.Introduced Strategy Pattern

Earlier the Game Play was happening directly and only between the human players. Now the assign countries phase , reinforcement phase and gameplay phase all are happening automatically. New strategies are also implemented such as Random Player Strategy, Aggressive Player Strategy, Benevolent Player Strategy , Cheater Player Strategy.

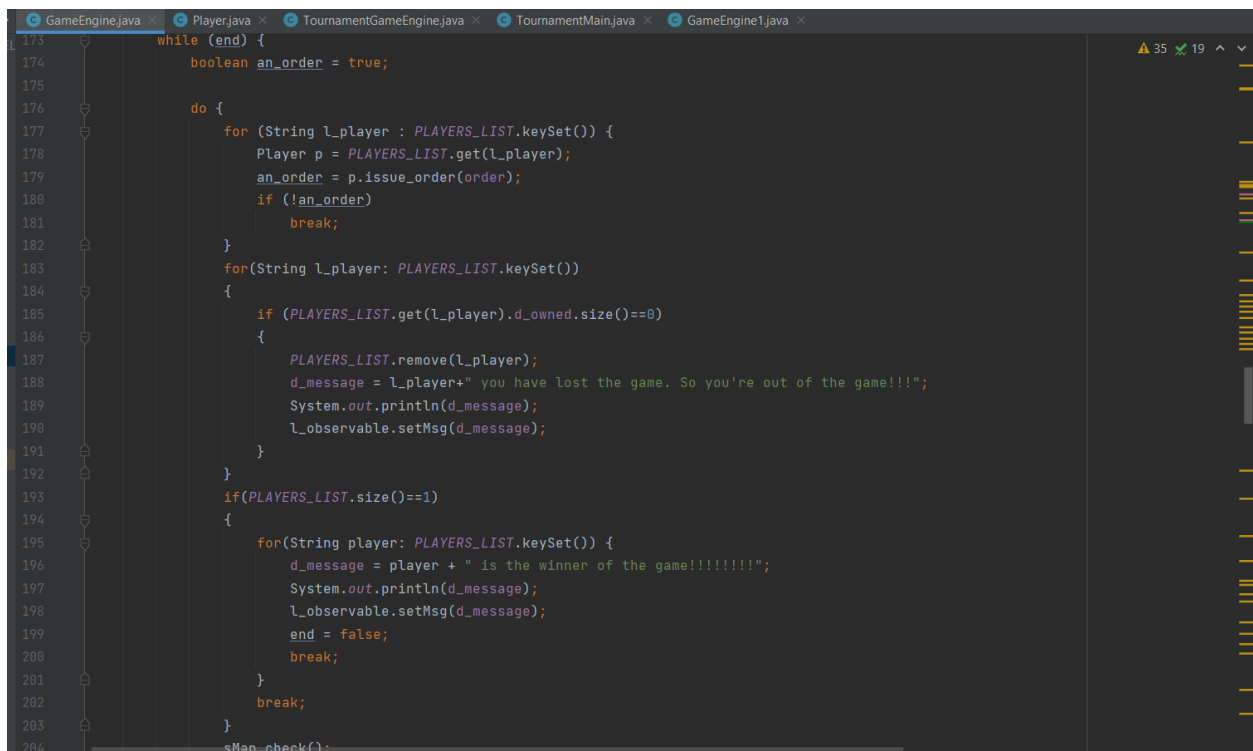
### ASSOCIATED TEST CASES.

1.**public void gameloop()**

2.**public void gameloop1()**

3.**public void checkStartUpPhase()**

4.**public void EndGametest()**



```
173 while (end) {
174     boolean an_order = true;
175
176     do {
177         for (String l_player : PLAYERS_LIST.keySet()) {
178             Player p = PLAYERS_LIST.get(l_player);
179             an_order = p.issue_order(order);
180             if (!an_order)
181                 break;
182         }
183         for(String l_player: PLAYERS_LIST.keySet())
184         {
185             if (PLAYERS_LIST.get(l_player).d_owned.size()==0)
186             {
187                 PLAYERS_LIST.remove(l_player);
188                 d_message = l_player+" you have lost the game. So you're out of the game!!!";
189                 System.out.println(d_message);
190                 l_observable.setMsg(d_message);
191             }
192         }
193         if(PLAYERS_LIST.size()==1)
194         {
195             for(String player: PLAYERS_LIST.keySet()) {
196                 d_message = player + " is the winner of the game!!!!!!!!!!";
197                 System.out.println(d_message);
198                 l_observable.setMsg(d_message);
199                 end = false;
200                 break;
201             }
202             break;
203         }
204     }
205     sMap.check();
206     return true;
207 }
```

```
public void executeAllOrders() {  
    System.out.println("=====BEGIN EXECUTING ALL ORDERS=====");  
    Order order;  
    boolean still_more_orders = false;  
    do {  
        still_more_orders = false;  
        for (String l_s : PLAYERS_LIST.keySet()) {  
            Player p = PLAYERS_LIST.get(l_s);  
            order = p.next_order();  
            if (order != null) {  
                still_more_orders = true;  
                order.Execute(p);  
            }  
        }  
    } while (still_more_orders);  
    System.out.println("=====END EXECUTING ALL ORDERS=====");  
}
```

## 2.Introduced Adapter Pattern

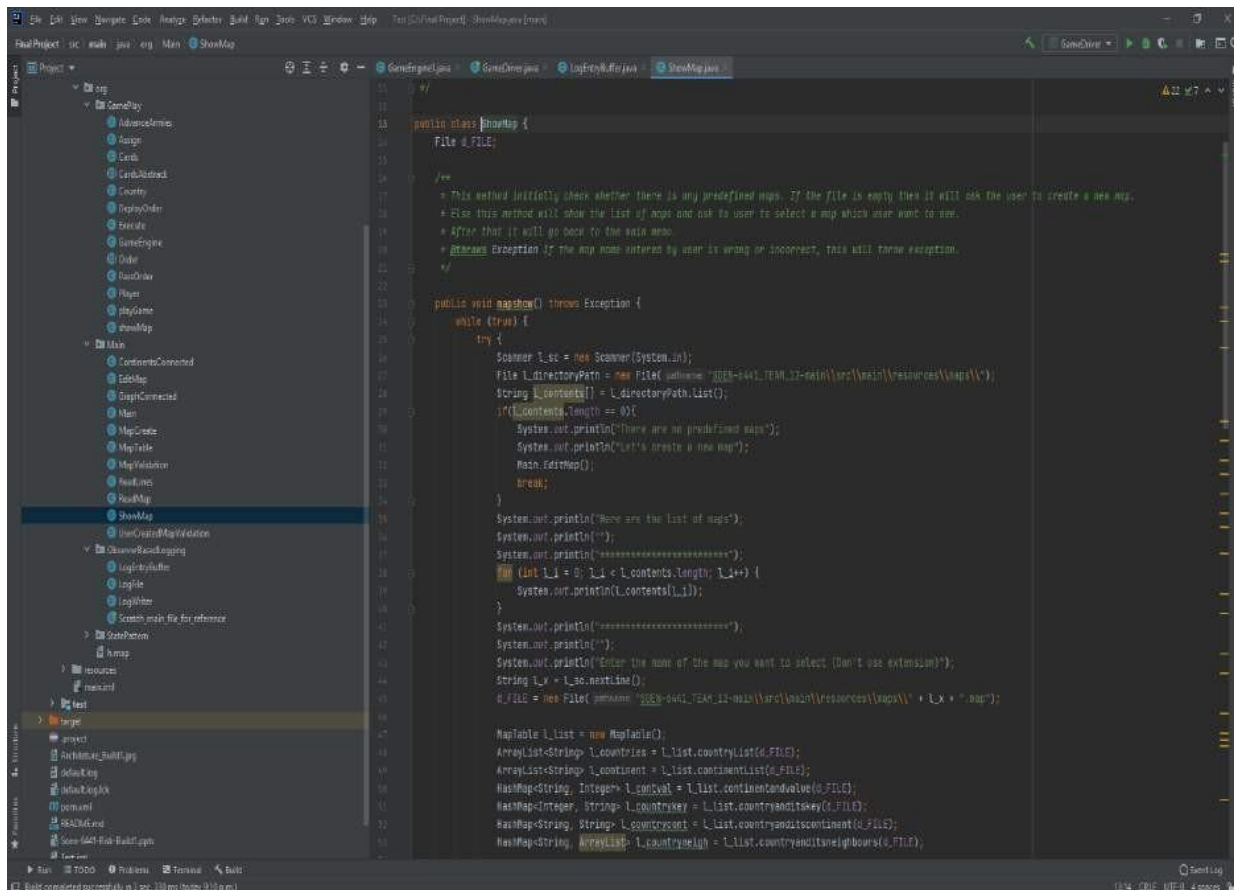
Previously, there was no adapter pattern, due to which loading of map was getting done in the ShowMap class only. But now we have introduced adapter pattern because of which the loading of map file is done separately in the Main class only.

### Associated Test Cases.

#### 1.ConquestGraphConnected()

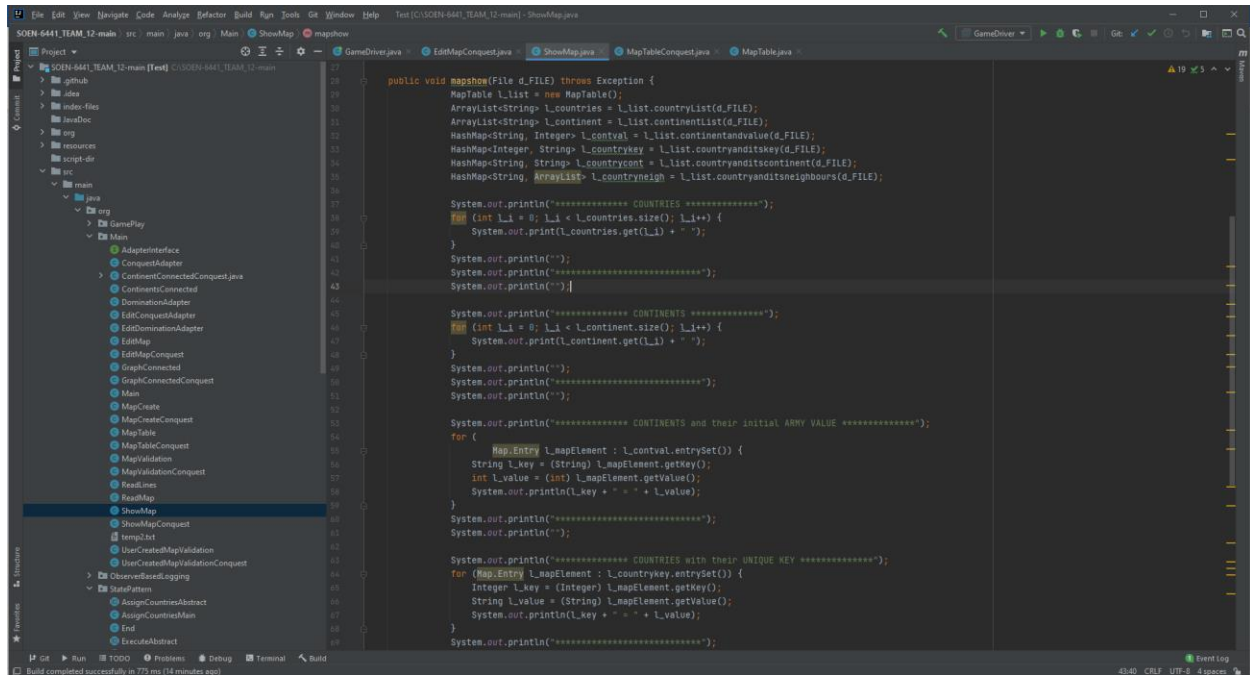
#### 2.ConquestGraphNotConnected()

Before:-



```
1  //
2
3  public class ShowMap {
4      File d.FILE;
5
6      /**
7       * This method initially check whether there is any predefined maps. If the file is empty then it will ask the user to create a new map.
8       * Else this method will show the list of maps and ask to user to select a map which user want to see.
9       * After that it will go back to the main menu.
10      * @throws Exception If the map name entered by user is wrong or incorrect, this will throw exception.
11      */
12
13      public void showMap() throws Exception {
14          while (true) {
15              try {
16                  Scanner l.sc = new Scanner(System.in);
17                  File l.directoryPath = new File(System.getProperty("user.dir") + "\\resources\\maps\\");
18                  String l.contents[] = l.directoryPath.list();
19                  if (l.contents.length == 0) {
20                      System.out.println("There are no predefined maps");
21                      System.out.println("Let's create a new map");
22                      Main.EditMap();
23                      break;
24                  }
25                  System.out.println("Here are the list of maps");
26                  System.out.println("");
27                  System.out.println("=====");
28                  for (int i = 0; i < l.contents.length; i++) {
29                      System.out.println(l.contents[i]);
30                  }
31                  System.out.println("=====");
32                  System.out.println("");
33                  System.out.println("Enter the name of the map you want to select (Don't use extension)");
34                  String l.x = l.sc.nextLine();
35                  d.FILE = new File(System.getProperty("user.dir") + "\\resources\\maps\\" + l.x + ".map");
36
37                  Mapable l.list = new Mapable();
38                  ArrayList<String> l.countries = l.list.countryList(d.FILE);
39                  ArrayList<String> l.continent = l.list.continentList(d.FILE);
40                  HashMap<String, Integer> l.contval = l.list.continentandvalue(d.FILE);
41                  HashMap<Integer, String> l.countrykey = l.list.countryanditskey(d.FILE);
42                  HashMap<String, String> l.countrycont = l.list.countryanditscontinent(d.FILE);
43                  HashMap<String, ArrayList> l.countryneigh = l.list.countryanditsneighbours(d.FILE);
44              } catch (Exception e) {
45                  e.printStackTrace();
46              }
47          }
48      }
49  }
```

After:-



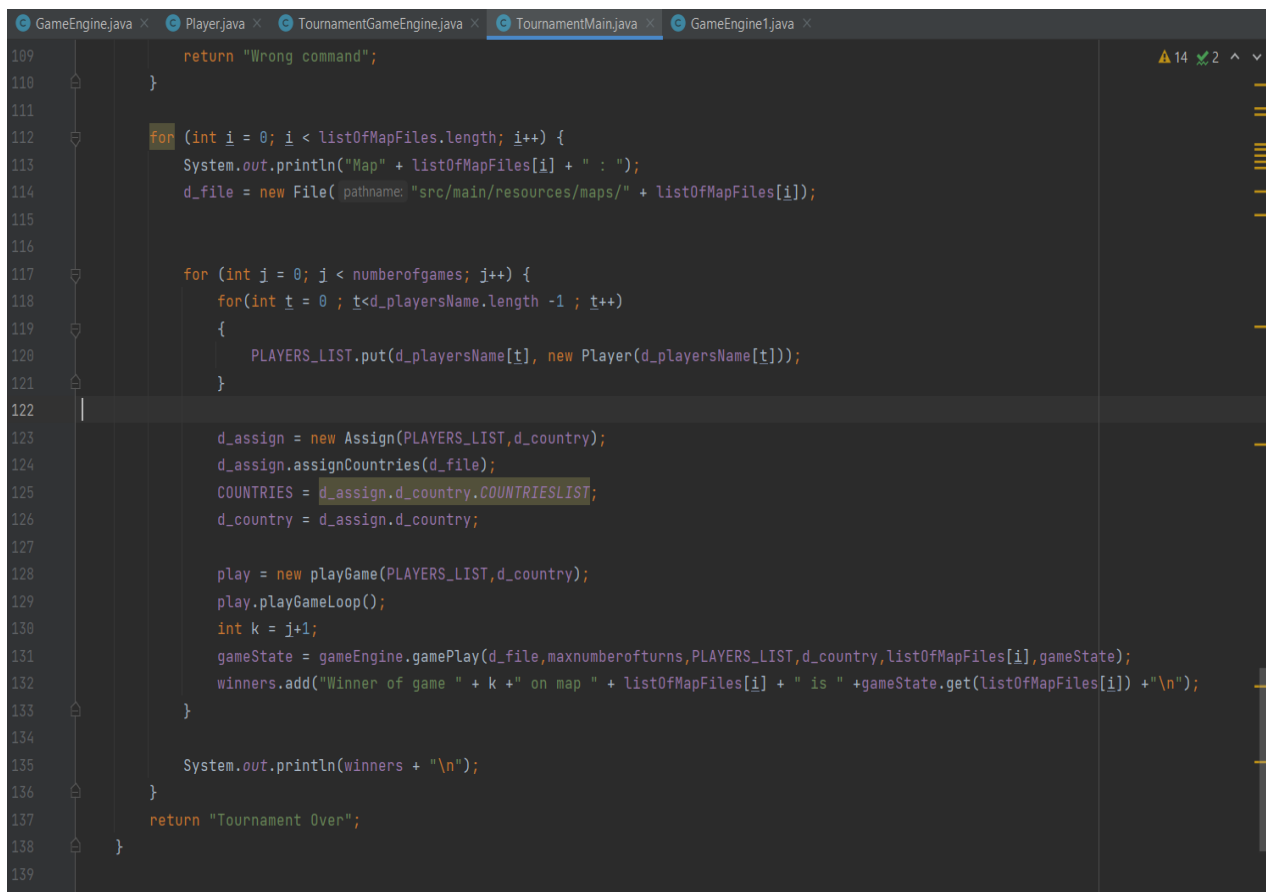
```
27 public void mapshon(File d_FILE) throws Exception {
28     MapTable l_list = new MapTable();
29     ArrayList<String> l_countries = l_list.countryList(d_FILE);
30     ArrayList<String> l_continent = l_list.continentList(d_FILE);
31     HashMap<String, Integer> l_contval = l_list.continentandvalue(d_FILE);
32     HashMap<String, String> l_countrykey = l_list.countryanditskey(d_FILE);
33     HashMap<String, String> l_countrycont = l_list.countryanditscontinent(d_FILE);
34     HashMap<String, ArrayList> l_countryneigh = l_list.countryanditsneighbours(d_FILE);
35
36     System.out.println("***** COUNTRIES *****");
37     for (int l_i = 0; l_i < l_countries.size(); l_i++) {
38         System.out.print(l_countries.get(l_i) + " ");
39     }
40     System.out.println("");
41     System.out.println("*****");
42     System.out.println("");
43
44     System.out.println("***** CONTINENTS *****");
45     for (int l_i = 0; l_i < l_continent.size(); l_i++) {
46         System.out.print(l_continent.get(l_i) + " ");
47     }
48     System.out.println("");
49     System.out.println("*****");
50     System.out.println("");
51
52     System.out.println("***** CONTINENTS and their initial ARMY VALUE *****");
53     for (
54         Map.Entry l_mapElement : l_contval.entrySet() {
55             String l_key = (String) l_mapElement.getKey();
56             int l_value = (int) l_mapElement.getValue();
57             System.out.println(l_key + " = " + l_value);
58         }
59     )
60     System.out.println("*****");
61     System.out.println("");
62
63     System.out.println("***** COUNTRIES with their UNIQUE KEY *****");
64     for (Map.Entry l_mapElement : l_countrykey.entrySet()) {
65         Integer l_key = (Integer) l_mapElement.getKey();
66         String l_value = (String) l_mapElement.getValue();
67         System.out.println(l_key + " = " + l_value);
68     }
69     System.out.println("*****");
70 }
```

## 3.Introduced Tournament Mode

Earlier only single Game mode was present ,now in Build 3 we have introduced Tournament Mode in this one can play multiple number of games on multiple numbers of maps.

### Associated Test Cases

1. runtestCompletefail()
2. runMapTest()
3. runTestPlayer()
4. runTestGame()
5. runTestTurns()

A screenshot of an IDE with several tabs open: GameEngine.java, Player.java, TournamentGameEngine.java, TournamentMain.java (selected), and GameEngine1.java. The code in TournamentMain.java is shown, starting from line 109. It includes a 'return "Wrong command";' statement, followed by a loop over 'listOfMapFiles'. Inside this loop, it prints the map name, creates a File object, and then enters a nested loop over 'numberOfGames'. This nested loop contains a loop over 'd\_playersName' to create a 'PLAYERS\_LIST' of 'Player' objects. After the nested loops, it creates an 'Assign' object, assigns countries to the file, and then creates a 'playGame' object. It then calls 'gameEngine.gamePlay' with various parameters, including the file, number of turns, players list, country, and map file. Finally, it adds the winner to a 'winners' list and prints the winners. The code ends with 'return "Tournament Over";'.

## 4.Introduced New Class to Check map validation of conquest maps.

In this build other than the domination worlds map we also have to play on conquest type maps and in order to play on them we also need to validate them so instead of using the old GraphConnected.java we have introduced new GraphConnectedConquest.java in order to handle the map validation of Conquest maps.

### Associated Test Cases.

#### 1.ConquestGraphConnected()

#### 2.ConquestGraphNotConnected()

```
ContinentConnectedConquest.java
30      * if both are equal the continents are connected and if not the not connected.
31      * @return boolean value whether a particulars maps continents are connected or not.
32      * @throws Exception
33      */
34      public boolean ifContinentsConnected() throws Exception {
35
36          MapTableConquest l_obj = new MapTableConquest();
37          ArrayList<Integer> l_ContinentAndKey = l_obj.continentsKeyConquest(d_file);
38          ArrayList<Integer> l_ContId = l_obj.CountriesContinentConquest(d_file);
39          Collections.sort(l_ContId);
40          GraphConnectedConquest l_check = new GraphConnectedConquest(d_file);
41
42          if (l_check.ifGraphConnected() == true) {
43
44              if (l_ContinentAndKey.equals(l_ContId)==true){
45                  return true;
46              }
47
48          }
49
50          return false;
51      }
52
53      public HashMap<Integer,ArrayList> ifContinentCountriesConnected() throws Exception {
54          MapTableConquest l_obj = new MapTableConquest();
55
56          ArrayList<Integer> copy = new ArrayList<>();
57          HashMap<Integer,ArrayList> nodes = new HashMap<>();
58          HashMap<Integer,Integer> l_connect = l_obj.contConquest(d_file);
```

```

93  /**
94   * This method finally checks the visited boolean array and sees if any value is false in it.
95   * If yes that means our graph is not connected and if no our graph is connected.
96   * @return boolean variable boolean which is simple true or false.
97   */
98   public boolean ifGraphConnected() {
99       int l_key_1 = d_countryNeigh.keySet().stream().findFirst().get();
100       int p_start = l_key_1;
101       dfs(p_start);
102       d_visited[0] = true;
103       for (int l_i = 0; l_i < d_realNodes.size(); l_i++) {
104           int l_check = d_realNodes.get(l_i);
105           boolean l_comp = d_visited[l_check];
106           if (l_comp == false) {
107               return false;
108           }
109       }
110       return true;
111   }
112
113
114   /**
115   * This function checks if each country is a connected sub graph.
116   * @return boolean whether the individual continent is connected or not.
117   * @throws Exception as we are using ConnectedContinent class Functions
118   */
119   public boolean ContinentsCheck() throws Exception {
120       ContinentsConnectedConquest d_obj = new ContinentsConnectedConquest(FILE);
121       HashMap<Integer, ArrayList> l_list = d_obj.ifContinentCountriesConnected();
122       for (Map.Entry<Integer, ArrayList> l_entry: l_list.entrySet()) {
123           HashMap<Integer, ArrayList> l_check = d_borders.CountryIDConquest(FILE, l_entry.getValue());

```

## 5.Reduced the number of phases as now the game is running automatically.

Earlier it was asked from the user to enter commands on the console to assign countries and also for the reinforcement phase now all that is happening automatically since the introduction of adapter pattern.

### ASSOCIATED TEST CASES.

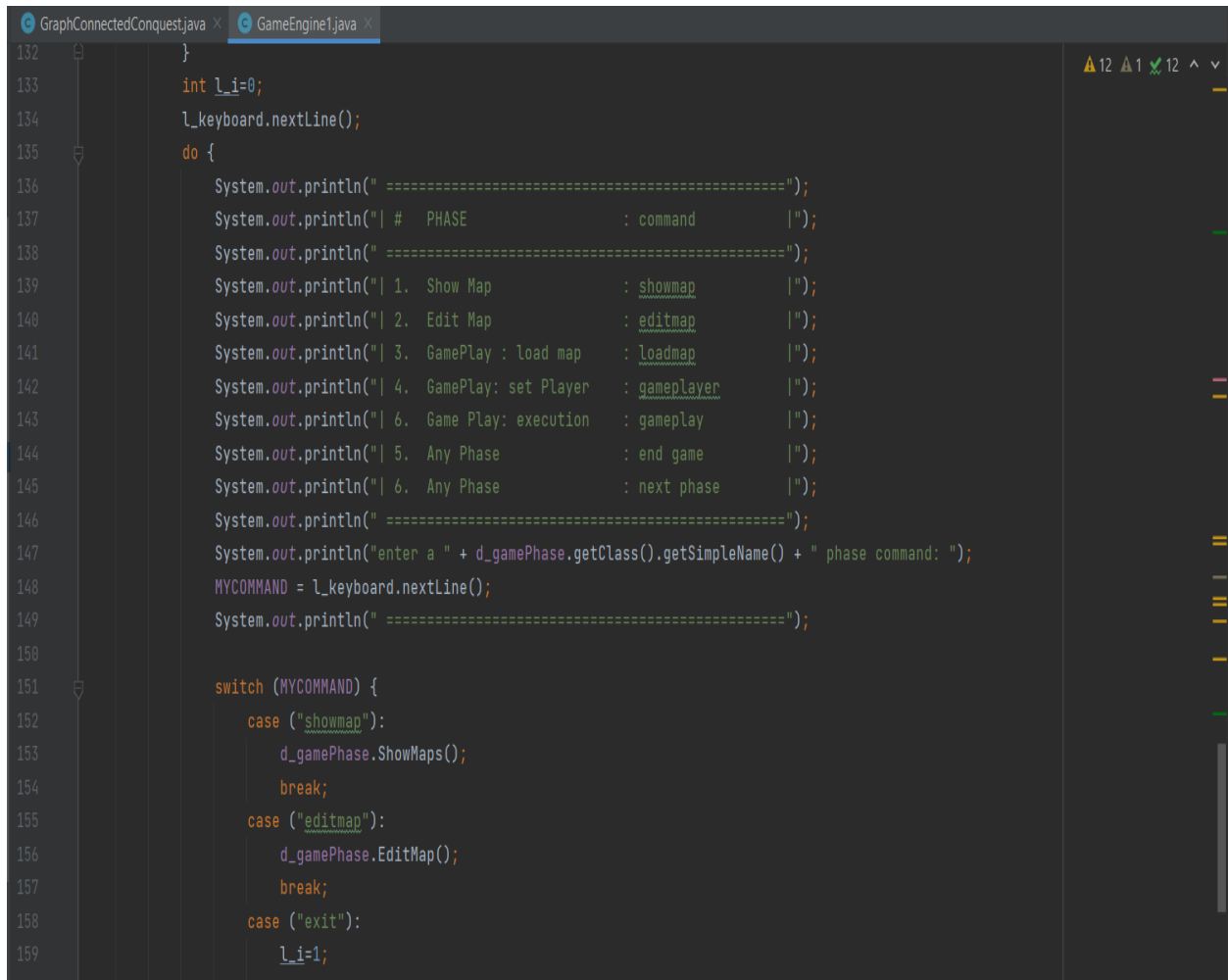
- 1.`public void gameloop()`
- 2.`public void gameloop1()`
- 3.`public void checkStartUpPhase()`
- 4.`public void EndGametest()`

Before:-

```
        System.out.println("Bye!");
        return;
    }
    int l_i=0;
    l_keyboard.nextLine();
    do {
        System.out.println(" =====");
        System.out.println("| #    PHASE                : command          |");
        System.out.println(" =====");
        System.out.println("| 1.  Show Map                : showmap           |");
        System.out.println("| 2.  Edit Map                : editmap           |");
        System.out.println("| 3.  GamePlay : load map     : loadmap           |");
        System.out.println("| 4.  GamePlay: set Player    : gameplayer        |");
        System.out.println("| 5.  Play:PlaySetup          : assign countries   |");
        System.out.println("| 6.  Play:Issue Order        : order             |");
        System.out.println("| 7.  Play:Execute Order      : execute            |");
        System.out.println("| 7.  Play:mapshow            : mapshow           |");
        System.out.println("| 9.  Any Phase               : end game           |");
        System.out.println("| 10. Any Phase               : next phase          |");
        System.out.println(" =====");
```



After:-



```
132 }
133 int l_i=0;
134 l_keyboard.nextLine();
135 do {
136     System.out.println(" =====");
137     System.out.println("| #   PHASE           : command       |");
138     System.out.println(" =====");
139     System.out.println("| 1. Show Map         : showmap         |");
140     System.out.println("| 2. Edit Map         : editmap         |");
141     System.out.println("| 3. Game Play : load map : loadmap         |");
142     System.out.println("| 4. Game Play: set Player : gameplayer   |");
143     System.out.println("| 6. Game Play: execution : gameplay     |");
144     System.out.println("| 5. Any Phase       : end game       |");
145     System.out.println("| 6. Any Phase       : next phase     |");
146     System.out.println(" =====");
147     System.out.println("enter a " + d_gamePhase.getClass().getSimpleName() + " phase command: ");
148     MYCOMMAND = l_keyboard.nextLine();
149     System.out.println(" =====");
150
151     switch (MYCOMMAND) {
152         case ("showmap"):
153             d_gamePhase.ShowMaps();
154             break;
155         case ("editmap"):
156             d_gamePhase.EditMap();
157             break;
158         case ("exit"):
159             l_i=1;
```

## 6.Code Repitition in Editmap class

- There is validation call part in the EditMap class which is called thrice in the same class.
- We could have refactored this one too, but this is not an important part which we need to do so.
- Also, this is making the understandability of code much easier.

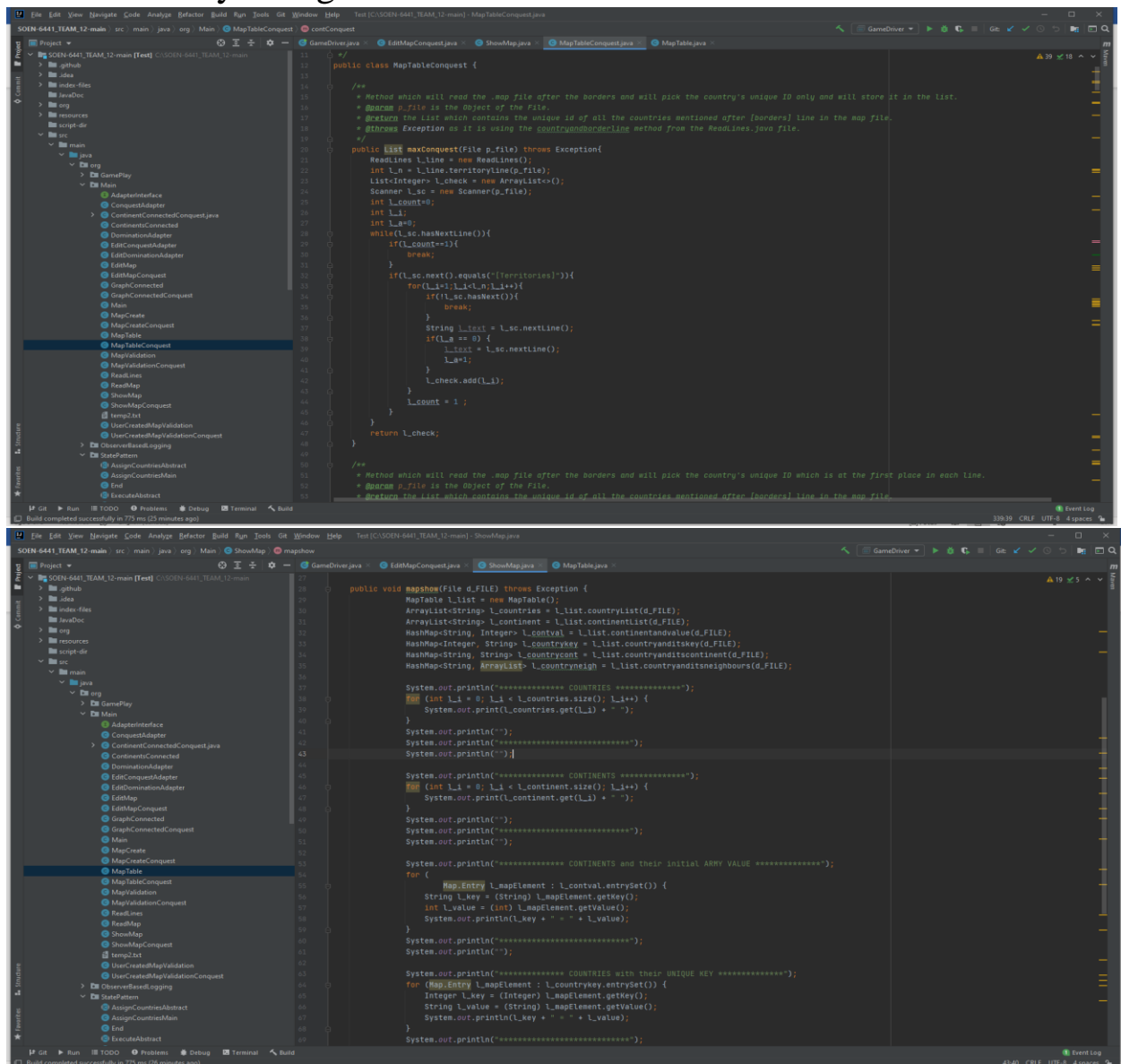
```
File l_file2 = new File(l_address);
l_flag_error=0;
while(true) {
    try {
        System.out.println("Enter validatemap command");
        String l_reply = d_SC.nextLine();
        if (l_reply.equalsIgnoreCase( anotherString: "validatemap")) {
            MapValidation l_validate = new MapValidation();
            l_validate.mapValidate(l_file2, l_countries, l_continent, l_cont_val, l_country_key, l_country_cont
            break;
        }
        else{
            System.out.println("Enter valid command");
        }
    } catch (Exception p_e) {
        p_e.printStackTrace();
    }
}
```

```
File l_file3 = new File(l_address);
l_flag_error = 0;

while(true) {
    try {
        System.out.println("Enter validatemap command");
        String l_reply = d_SC.nextLine();
        if (l_reply.equalsIgnoreCase( anotherString: "validatemap")) {
            MapValidation l_validate = new MapValidation();
            l_validate.mapValidate(l_file3, l_countries, l_continent, l_cont_val, l_country_key, l_country_cont
            break;
        }
        else{
            System.out.println("Enter valid command");
        }
    } catch (Exception p_e) {
        p_e.printStackTrace();
    }
}
```

## 7. We could have used single MapTable.java class.

- In MapEditor phase we are using a class named MapTable to fetch information about the map file like the countries list, continents list.
- Previously, there was only one class but due to Adapter pattern now there are two map files i.e., domination map file and conquest map file. So, we have introduced two classes for that one is MapTable class and the other one is MapTableConquest. We could have used only a single class.



The image displays two screenshots of an IDE (IntelliJ IDEA) showing Java code. The top screenshot shows the `MapTableConquest` class, and the bottom screenshot shows the `MapTable` class.

**Top Screenshot: `MapTableConquest` class**

```
17  public class MapTableConquest {
18
19      // Method which will read the .map file after the borders and will pick the country's unique ID only and will store it in the list.
20      // Because p_file is the object of the File.
21      // Returns the list which contains the unique id of all the countries mentioned after [borders] line in the map file.
22      // Throws Exception as it is using the countryNumbering method from the ReadLines.java file.
23
24      public List maxConquest(File p_file) throws Exception {
25          ReadLines l_line = new ReadLines();
26          int Lc = l_line.territoryLine(p_file);
27          List<Integer> L_check = new ArrayList<>();
28          Scanner L_sc = new Scanner(p_file);
29
30          int L_count=0;
31          int L_i=0;
32          int L_a=0;
33          while(L_sc.hasNextLine()){
34              if(L_count==1){
35                  break;
36              }
37              if(L_sc.next().equals("-territories-")){
38                  for(L_i=L_sc.next();L_i!="-";){
39                      if(L_sc.hasNextLine()){
40                          break;
41                      }
42                      String L_text = L_sc.nextLine();
43                      if(L_a == 0) {
44                          L_text = L_sc.nextLine();
45                          L_a=1;
46                      }
47                      L_check.add(L_i);
48                      L_count + 1 ;
49                  }
50              }
51              return L_check;
52          }
53      }
54      // Method which will read the .map file after the borders and will pick the country's unique ID which is at the first place in each line.
55      // Because p_file is the object of the File.
56      // Returns the list which contains the unique id of all the countries mentioned after [borders] line in the map file.
57  }
```

**Bottom Screenshot: `MapTable` class**

```
27  public void mapshow(File d_FILE) throws Exception {
28      MapTable L_list = new MapTable();
29      ArrayList<String> L_countries = L_list.countryList(d_FILE);
30      ArrayList<String> L_continent = L_list.continentList(d_FILE);
31      HashMap<String, Integer> L_contval = L_list.continentandvalue(d_FILE);
32      HashMap<Integer, String> L_countrykey = L_list.countryanditskey(d_FILE);
33      HashMap<String, String> L_countrycont = L_list.countryanditscontinent(d_FILE);
34      HashMap<String, ArrayList> L_countryneigh = L_list.countryanditsneighbours(d_FILE);
35
36      System.out.println("===== COUNTRIES =====");
37      for (int L_i = 0; L_i < L_countries.size(); L_i++) {
38          System.out.println(L_countries.get(L_i) + " ");
39      }
40      System.out.println("");
41      System.out.println("===== CONTINENTS =====");
42      for (int L_i = 0; L_i < L_continent.size(); L_i++) {
43          System.out.println(L_continent.get(L_i) + " ");
44      }
45      System.out.println("");
46      System.out.println("===== CONTINENTS and their initial ARMY VALUE =====");
47      for (Map.Entry L_mapElement : L_contval.entrySet()) {
48          String L_key = (String) L_mapElement.getKey();
49          int L_value = (int) L_mapElement.getValue();
50          System.out.println(L_key + " : " + L_value);
51      }
52      System.out.println("");
53      System.out.println("===== COUNTRIES with their UNIQUE KEY =====");
54      for (Map.Entry L_mapElement : L_countrykey.entrySet()) {
55          Integer L_key = (Integer) L_mapElement.getKey();
56          String L_value = (String) L_mapElement.getValue();
57          System.out.println(L_key + " : " + L_value);
58      }
59      System.out.println("");
60  }
```

## 8. Introduced new Gameplay phase in the game.

Since the introduction of the strategy pattern all the game has become automatic. So to do that we have introduced the new phase called gameplay phase once the player names has been selected then the user will time the gameplay command and the game will automatically run and at the end give us the winner.

```
GamePlayMain.java
1 package org.StatePattern;
2
3 import org.GamePlay.GameEngine;
4
5 public class GamePlayMain extends GamePlayAbstract{
6     GameEngine d_ge;
7     GameEngine1 d_ge1;
8
9     /**
10      * Constructor to initialize the values.
11      * @param p_ge is the Object of the class GameEngine1.
12      * @param p_ge1 is the Object of the class GameEngine
13      */
14     public GamePlayMain(GameEngine1 p_ge,GameEngine p_ge1) {
15         super(p_ge);
16         this.d_ge1 = p_ge;
17         this.d_ge = p_ge1;
18     }
19
20     /**
21      * Method to start the GameEngine.
22      */
23     @Override
24     public void gamePlay() { d_ge.gamePlay(d_ge.FILE); }
25
26
27
28     /**
29      * Method to change the Phase. If the size of the
30      */
31 }
```

## 9. Player class is modified.

Earlier in issue order functions we were simply issuing orders from the console now in issue order function players of respective strategies will give the order accordingly.

```
GamePlayMain.java x Player.java x
12 /**
13  *Player Class containing all the stats of the Player Class
14  */
15  public class Player implements Serializable {
16      private static final long serialVersionUID= 3710154222911855669L;
17      public ArrayList<Country> d_owned = new ArrayList<>();
18      public Queue<Order> d_orders = new LinkedList<>(); // Queue only
19      public Integer d_armiesNum;
20      public String d_name;
21      public int d_continentValue; // Continent Control Value ;
22      public ArrayList<String> d_cards = new ArrayList<>();
23      public ArrayList<String> d_negotiate = new ArrayList<>();
24      PlayerStrategy strategy;
25
26
27  /**
28   * Parameterised Constructor to initilize Player Object by name .
29   * @param p_name Player Name
30   */
31  public Player(String p_name) { this.d_name = p_name; }
32
33
34
35  public void setStrategy(PlayerStrategy p_strat) { strategy = p_strat; }
36
37
38
39  /**
```

```
/**
 *function to issue order .
 * @param order issue order object
 */
public boolean issue_order(Order order)
{
    order = strategy.createOrder();
    if (order != null) {
        d_orders.add(order);
        return true;
    }
    return false;
}
```

## 10. playGame.java is Modified

Earlier in maingameloop() function we were simply adding the orders to the orders list now here we are returning orders to the player class and then respective player is saving the orders in their list.

```
System.out.println("**Deploy**\n**Advance**\n**Cards**\n    -Bomb\n    -Blockade\n    -AirLift\n    -Negotiate\n**Pass**\n**Sho");
System.out.println("*****");
Scanner l_sc = new Scanner(System.in);
String l_command = l_sc.nextLine();
String l_commandSplit[] = l_command.split(regex: " ");
if (l_commandSplit[0].equalsIgnoreCase( anotherString: "deploy")) {
    if (l_commandSplit.length == 3) {
        String l_countryId = l_commandSplit[1];
        if (d_country.COUNTRIESLIST.containsKey(l_countryId)) {
            String l_regex = "\\d+";
            if (l_commandSplit[2].matches(l_regex)) {
                int l_armiesToPlace = Integer.parseInt(l_commandSplit[2]);
                if (l_armiesToPlace > 0) {
                    l_flag_1 = false;
                    return new DeployOrder(l_countryId, l_armiesToPlace, d_country);
                } else {
                    System.out.println("Negative Army count not allowed");
                    continue;
                }
            } else {
                System.out.println("The army number should be an integer");
                continue;
            }
        } else {
            System.out.println("The country that you entered doesn't exist in the Map");
            continue;
        }
    } else {
        System.out.println("Your deploy command should be like 'deploy countryName armyToPlace'\n");
    }
}
```

## 11.In Cards we can encapsulate each card functionality into a function and simply call it.

- We have simple just implemented the functionality of each card directly in the respective switch cases.
- We can create functions for each card and then simply call it under the switch case.

```
case "blockade":
    if(p_p.d_cards.contains("BLOCKADE") && p_p.d_owned.contains(d_country.COUNTRIESLIST.get(d_CountryID))) {
        d_country.COUNTRIESLIST.get(d_CountryID).d_numOfArmiesPlaced -= 3;
        p_p.d_owned.remove(d_country.COUNTRIESLIST.get(d_CountryID));
        d_country.COUNTRIESLIST.get(d_CountryID).d_owner = "Neutral";
        l_message = d_CountryID + " has been neutralized , now nobody owns this country";
        System.out.println(l_message);
        l_observable.setMsg(l_message);
        int l_indexCardName=-1;
        for(String l_cardName: p_p.d_cards)
        {
            l_indexCardName++;
            if(l_cardName.equalsIgnoreCase( anotherString: "BLOCKADE"))
            {
                p_p.d_cards.remove(l_indexCardName);
                break;
            }
        }
    }
    else{
        l_message = "You don't have the BLOCKADE card.";
        System.out.println(l_message);
        l_observable.setMsg(p_p+" "+l_message);
    }
    break;
case "airLift":
    if(p_p.d_cards.contains("AIRLIFT")) {
        if (!p_p.d_negotiate.contains(d_country.COUNTRIESLIST.get(d_CountryTo).d_owner)) {
            if (d_country.COUNTRIESLIST.get(d_CountryFrom).d_numOfArmiesPlaced >= d_armiesToAirLift) {
                d_obj = new AdvanceArmies(d_CountryFrom, d_CountryTo, d_armiesToAirLift, d_country, d_card);
                d_obj.Execute(p_p);
            }
        }
    }
}
```

## 12.Invalid Command Handling can be encapsulated inside a function in playGame class.

- No when the user types command ,to validate this command all the handling is done using if-else statements.

We can do this handling inside a function and simply call it at the appropriate place

```
} else if (l_commandSplit[0].equalsIgnoreCase( anotherString: "bomb")) {
    if (l_commandSplit.length == 2) {
        String l_countryId = l_commandSplit[1];
        if (d_country.COUNTRIESLIST.containsKey(l_countryId)) {
            if (!l_p.d_owned.contains(d_country.COUNTRIESLIST.get(l_countryId))) {
                l_flag_1 = false;
                Cards l_card = new Cards(l_countryId, l_commandSplit[0], d_country);
                l_p.issue_order(l_card);
            } else {
                System.out.println("Cannot Bomb its own country");
                continue;
            }
        } else {
            System.out.println("Not a Valid Country");
            continue;
        }
    } else {
        System.out.println("Not a Valid Command");
        continue;
    }
} else if (l_commandSplit[0].equalsIgnoreCase( anotherString: "advance")) {
    if (l_commandSplit.length == 4) {
        String l_country_from = l_commandSplit[1];
        String l_country_to = l_commandSplit[2];
        String l_armies = l_commandSplit[3];
        String l_card = "empty";
        int l_armiesToAdvance = Integer.parseInt(l_armies);
        if (d_country.COUNTRIESLIST.containsKey(l_country_from) && d_country.COUNTRIESLIST.containsKey(l_country_to)) {
            if (l_p.d_owned.contains(d_country.COUNTRIESLIST.get(l_country_from))) {
                if (d_country.COUNTRIESLIST.get(l_country_from).d_neighbours.contains(l_country_to)) {
                    if (l_p.d_armiesNum >= l_armiesToAdvance) {
```



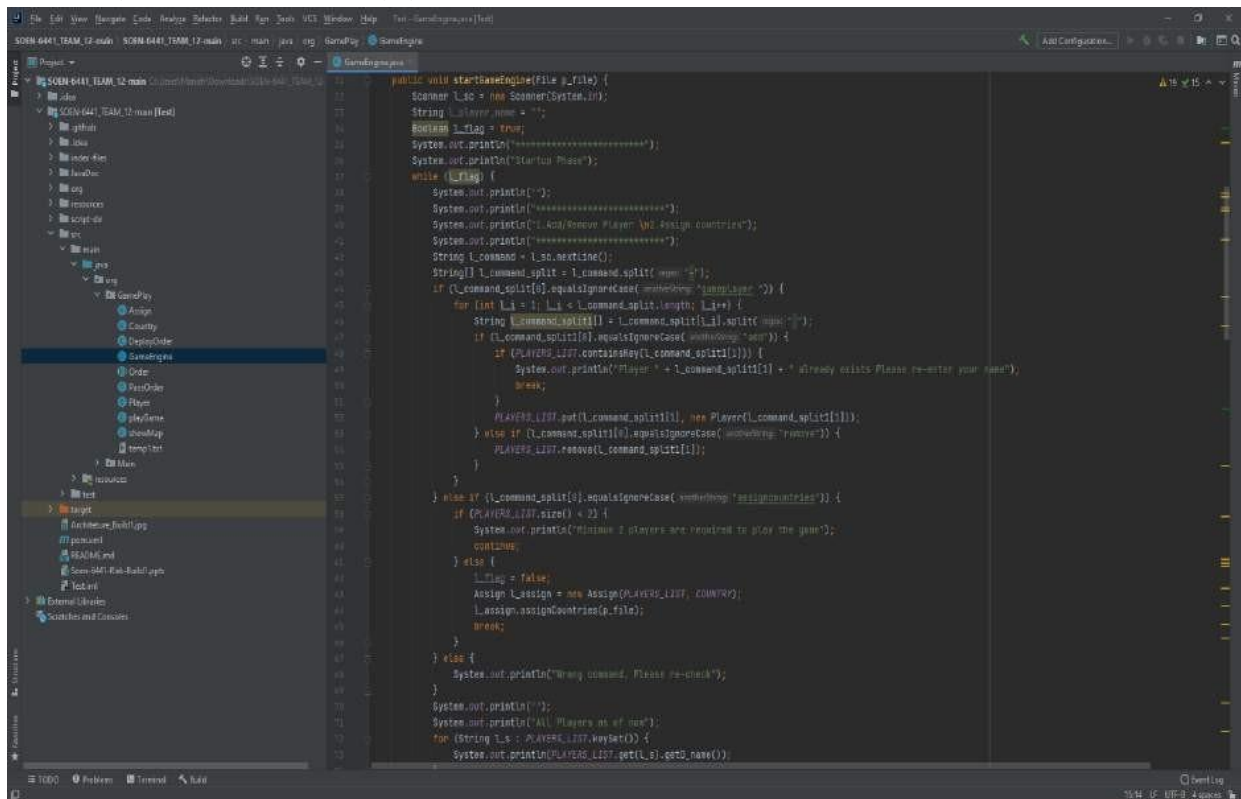
## 13.Code repetition in continent connectivity check

- In Graph Connected class for countries connectivity there is dfs() function and finally checking the connectivity ifGraphConnected() function is there.
- In ContinentCheck both above functions can be used but instead whole code is written from scratch.
- This can be avoided by making dfs() and ifGraphConnected() function generalized.

```
GraphConnected.java
120 public boolean ContinentsCheck() throws Exception {
121     ContinentsConnected d_obj = new ContinentsConnected(FILE);
122     HashMap<Integer, ArrayList> l_list = d_obj.ifContinentCountriesConnected();
123     for(Map.Entry<Integer, ArrayList> l_entry: l_list.entrySet()) {
124         HashMap<Integer, ArrayList> l_check = d_obj.CountryNeighbours(FILE, l_entry.getValue());
125         List<Integer> l_nodes = l_entry.getValue();
126         ArrayList<Integer> l_empty = new ArrayList<>();
127         if(l_nodes.size() == 1){
128             l_check.put(l_nodes.get(0), l_empty);
129         }
130         int l_max = Collections.max(l_nodes);
131         d_scan = new boolean[l_max + 1];
132         boolean [] l_traversed = new boolean[l_max + 1];
133         int l_key_1 = l_check.keySet().stream().findFirst().get();
134         int p_start = l_key_1;
135         d_stack.push(p_start);
136         l_traversed[p_start] = true;
137         d_country_index = l_nodes;
138         d_scan[0] = true;
139         for (int l_i = 0; l_i < d_country_index.size(); l_i++){
140             d_scan[d_country_index.get(l_i)] = true;
141         }
142         for (int l_j = 0; l_j < l_nodes.size(); l_j++){
143             if(l_check.containsKey(l_nodes.get(l_j)) == false){
144                 int l_note = l_nodes.get(l_j);
145                 l_check.put(l_note, l_empty);
146             }
147         }
148         int l_count = 0;
149         while (true) {
150             if (d_stack.isEmpty() == true) {
151                 break;
152             }
153         }
154     }
155 }
```

## 14.Changes in the GameEngine class

- We have broken the continuity of the game to introduce the state pattern in the game.
- Previously, there was a loop to add player and assign countries together but now we have now broken this continuity so as to create different phases for every part.



```
public void startBaseEngine(File p_file) {
    Scanner L_sc = new Scanner(System.in);
    String L_player_name = "";
    boolean L_flag = true;
    System.out.println("*****");
    System.out.println("Start-up Phase");
    while (L_flag) {
        System.out.println("-");
        System.out.println("*****");
        System.out.println("1.Add/Remove Player (Wl Assign countries);");
        System.out.println("*****");
        String L_command = L_sc.nextline();
        String[] L_command_split = L_command.split(" ");
        if (L_command_split[0].equalsIgnoreCase("addplayer")) {
            for (int Li = 1; Li < L_command_split.length; Li++) {
                String L_command_split1[] = L_command_split[Li].split(" ");
                if (L_command_split1[0].equalsIgnoreCase("new")) {
                    if (PLAYERS_LIST.contains(L_command_split1[1])) {
                        System.out.println("Player " + L_command_split1[1] + " already exists Please re-enter your name");
                        break;
                    }
                    PLAYERS_LIST.add(L_command_split1[1], new Player(L_command_split1[1]));
                } else if (L_command_split1[0].equalsIgnoreCase("remove")) {
                    PLAYERS_LIST.remove(L_command_split1[1]);
                }
            }
        } else if (L_command_split[0].equalsIgnoreCase("assigncountries")) {
            if (PLAYERS_LIST.size() < 2) {
                System.out.println("Minimum 2 players are required to play the game");
                continue;
            } else {
                L_flag = false;
                Assign L_assign = new Assign(PLAYERS_LIST, COUNTRY);
                L_assign.assignCountries(p_file);
                break;
            }
        } else {
            System.out.println("Wrong command, Please re-check");
        }
        System.out.println("-");
        System.out.println("All Players are of new");
        for (String Li : PLAYERS_LIST.keySet()) {
            System.out.println(PLAYERS_LIST.get(Li).getL_name());
        }
    }
}
```

## 15.Add Card Function is never used in Player Class

- add\_card() function inside player class is never used.
- Instead ArrayList d\_cards is directly accessed in various parts of our program.
- Either we can remove this function or we can use this function in order to access d\_cards arraylist where it is accessed in our code.

```
/**  
 * Method to add the cards to the list.  
 * @param cards is the Name of the card that player has won.  
 */  
public void add_card(String cards) { this.d_cards.add(cards); }  
/**
```