Refactoring

Sr No.	File Name	Line Number	Target	Description
1.	**Multiple Files**	**-**	**Strategy Pattern**	**Strategy Pattern Implemented**
2.	**Multiple Files**	**-**	**Adapter Pattern**	**Adapter Pattern Implemented**
3.	**Multiple Files**	**-**	**Changed order response and set appropriate message**	**Display output message based on command entered by user**
4.	**User Interface**	**-**	**Game Engine UI**	**Performed some changes in user interface to enhance user experience**
5.	**AdvanceOrder**	**50**	**boolean l_flag = true;**	**variable never used**
6.	AdvanceOrder	79,80,84, 85	l_countryTo.setD_noOfArmies(l_toArmies); l_countryfrom.setD_noOfArmies(l_fromArmies);	Redundant statement in if and else if so, move it down
7.	Deployorder Class	52,71	int l_getArmy = l_country.getD_noOfArmies();	2 conditions assigned multiple times (if and else if)
8.	GameConfigServiceImpl	53,57	String l_showMapOfCountris	Variable assigned multiple times for if and else condition
9.	GameConfigServiceImpl	70	continue	last statement of if so no need of it
10.	GameConfigServiceImpl	218	int l_i = 0, l_j = 0;	No need of this as in next line it automatically set to zero because of for loop
11.	MapHandlingImpl	78,83	l_isValid	value already assigned to false
12.	MapHandlingImpl	435 to 438	if statement	simplify if statement
13.	MapHandlingImpl	545	if (!l_continentList.contains(l_country.getD_continentIndex()))	loop can be break once the condition is fulfilled
14.	MapHandlingImpl	688	if (l_neighbour.equals(p_countryName))	loop can be break once the condition is fulfilled
15.	**MapHandlingImpl**	**1206**	**private int getCountryIndexByCountryName**	**JAVADOC Method parameter description is missing**
16.	MapHandlingImpl	506,509	l_showMapIn2D = l_showMapIn2D +	Variable assigned mjultiple times for if and else condition

			l_stringFrmat;	
17.	OrderProcessingImpl	55	Field l_player = null;	Variable never used
18.	OrderProcessingImpl	91,94	boolean l_invoke = false;	never used
19.	ExecuteOrderPhase	52	l_isueOrderPhase.d_gameData = (GameData) d_gameData;	No need of casting

Rational explanation

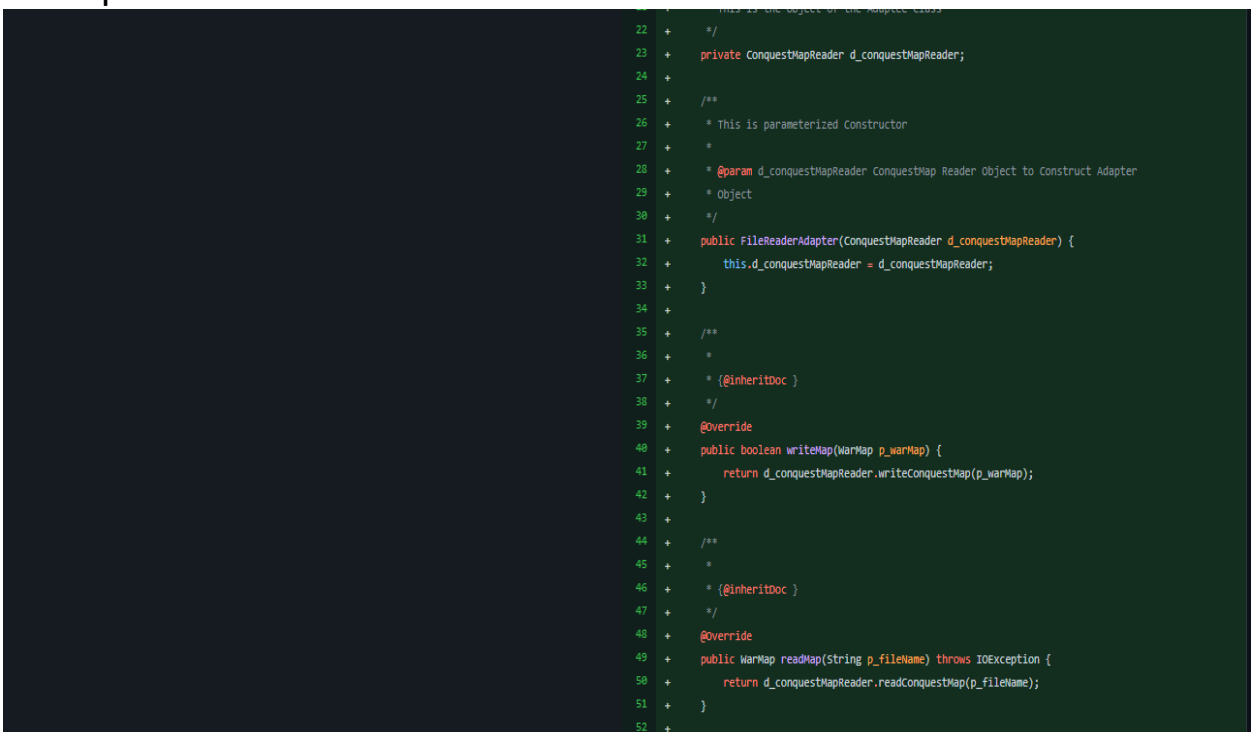
- Adapter Pattern – This design pattern allows using an interface of an existing class to be used as another interface and make it easy to work with other classes by using existing classes without modifying their source code.
- Strategy Pattern – This design pattern allows us to enable selected algorithm at runtime by providing family of algorithms to use at runtime instead of implementing a single algorithm
- Changed order response and set appropriate message – Here, instead of providing generalize response of command, we implemented specific command response so, and it will enhance the understandability.
- User Interface – For better user view, we modified the User Interface of Game Engine Phase.
- Removed unused variable - For better code readability as well as to reduce memory utilization.
- Other File Changes - We focused on necessary changes, which was crucial and required more attention as quickly as possible. So, we neglected these minor changes.

Actual refactoring targets

Refactoring Target	Test	Purpose
Adapter Pattern	testForReadMapFromFile() testForWriteMapToFile()	To allow to use an interface of an existing class to be used as another interface
Strategy Pattern	testAggressiveStrategy() testCheaterStrategy() testBenevolantStrategy() testRandomStrategy()	To enable selected algorithm at runtime by providing family of algorithms
Changed order response and set appropriate message	testAirliftCommand() testAdvanceCommand() testNegotiateCommand() testBlockadeCommand()	To provide specific response of command order instead of generalize response
User Interface	-	To enhance the user view of the Game Engine Phase
Remove unused variables	testAdvanceCommand() testAdvanceCommandWithOpponentCountry()	Remove unused variables

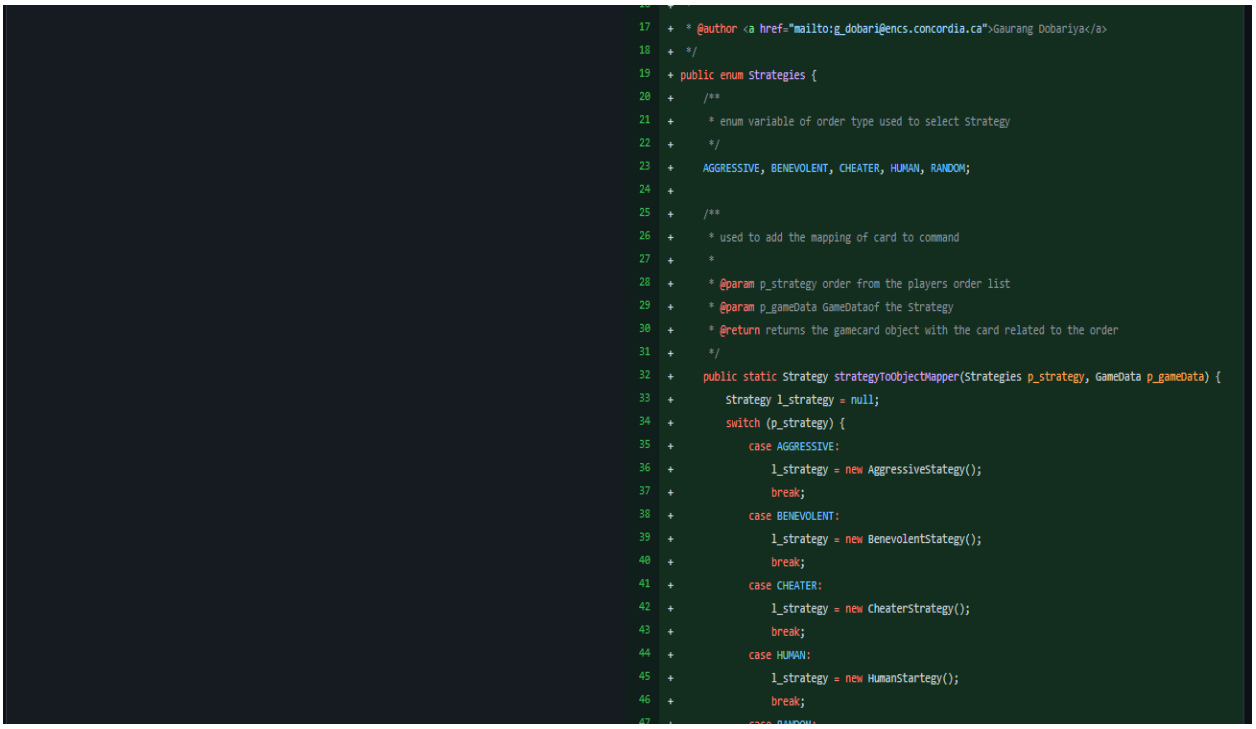
Before/After depiction

- Adapter Pattern -



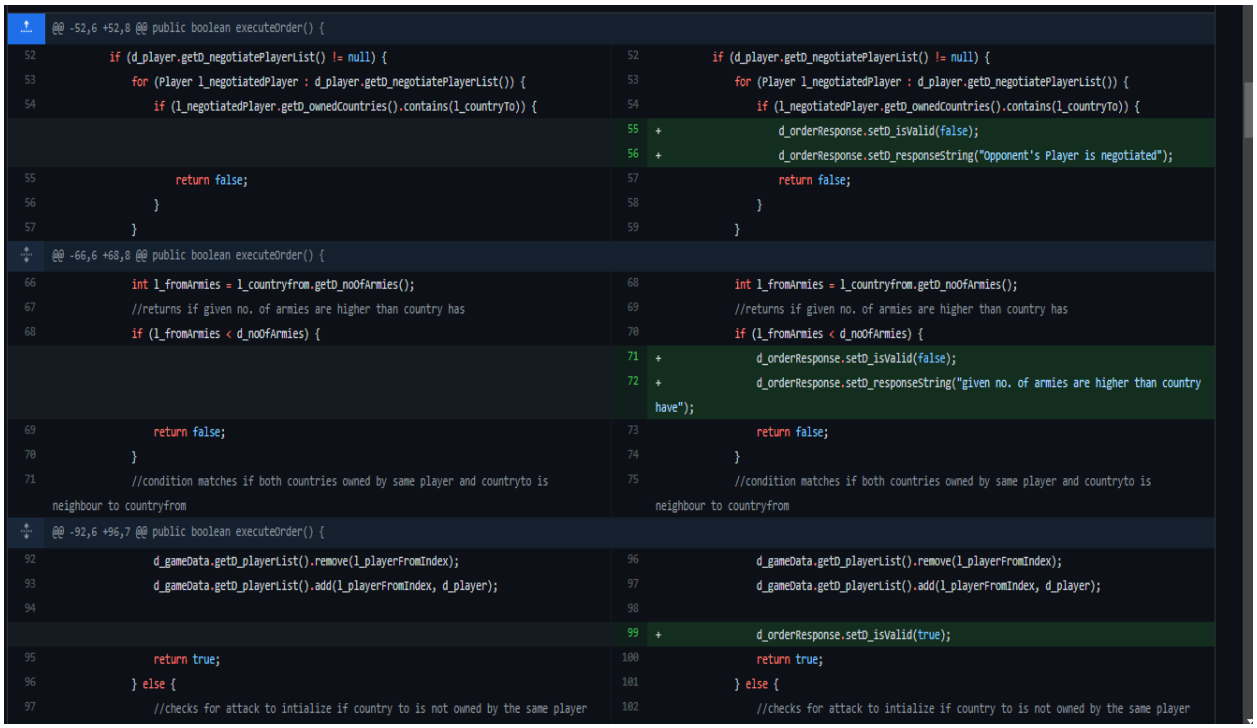
```
+     */
+    private ConquestMapReader d_conquestMapReader;
+
+    /**
+     * This is parameterized Constructor
+     *
+     * @param d_conquestMapReader ConquestMap Reader Object to Construct Adapter
+     * Object
+     */
+    public FileReaderAdapter(ConquestMapReader d_conquestMapReader) {
+        this.d_conquestMapReader = d_conquestMapReader;
+    }
+
+    /**
+     *
+     * {@inheritDoc }
+     */
+    @Override
+    public boolean writeMap(WarMap p_warMap) {
+        return d_conquestMapReader.writeConquestMap(p_warMap);
+    }
+
+    /**
+     *
+     * {@inheritDoc }
+     */
+    @Override
+    public WarMap readMap(String p_fileName) throws IOException {
+        return d_conquestMapReader.readConquestMap(p_fileName);
+    }
```

- Strategy Pattern –



```
+ * @author <a href="mailto:g_dobari@encs.concordia.ca">Gaurang Dobariya</a>
+ */
+public enum Strategies {
+    /**
+     * enum variable of order type used to select Strategy
+     */
+    AGGRESSIVE, BENEVOLENT, CHEATER, HUMAN, RANDOM;
+
+    /**
+     * used to add the mapping of card to command
+     *
+     * @param p_strategy order from the players order list
+     * @param p_gameData GameDataof the Strategy
+     * @return returns the gamecard object with the card related to the order
+     */
+    public static Strategy strategyToObjectMapper(Strategies p_strategy, GameData p_gameData) {
+        Strategy l_strategy = null;
+        switch (p_strategy) {
+            case AGGRESSIVE:
+                l_strategy = new AggressiveStategy();
+                break;
+            case BENEVOLENT:
+                l_strategy = new BenevolentStategy();
+                break;
+            case CHEATER:
+                l_strategy = new CheaterStrategy();
+                break;
+            case HUMAN:
+                l_strategy = new HumanStartegy();
+                break;
```

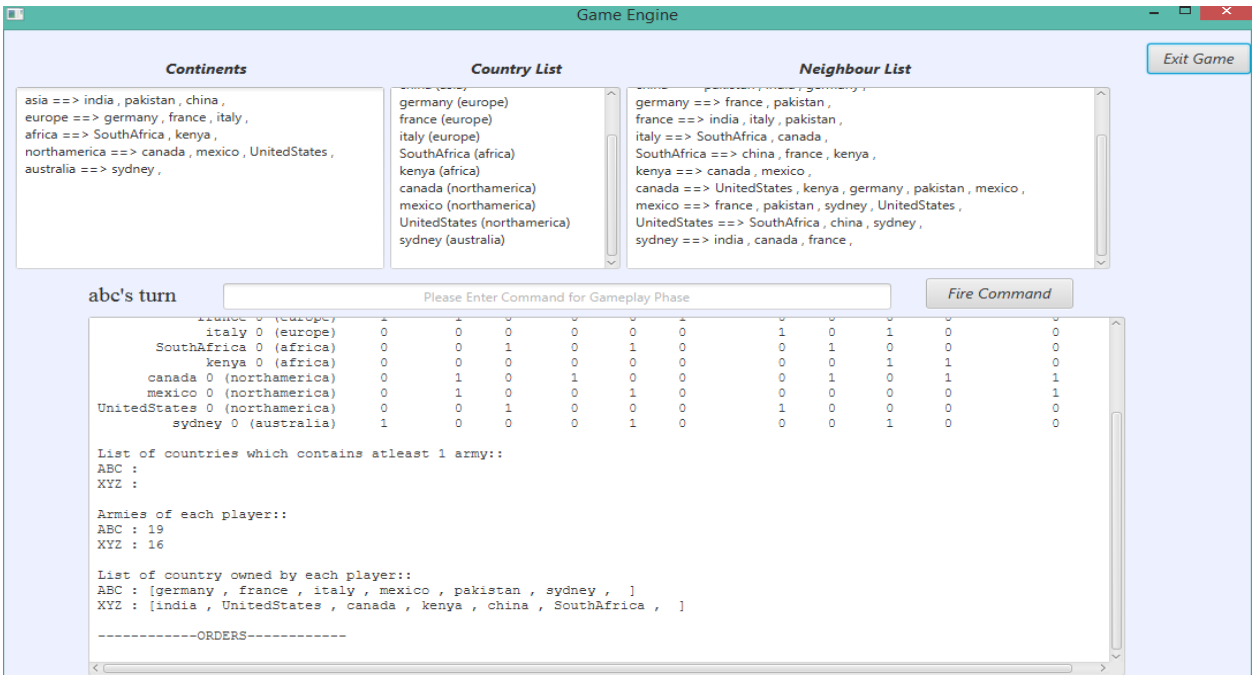
- Changed order response and set appropriate message –



```
@@ -52,6 +52,8 @@ public boolean executeOrder() {
            if (d_player.getD_negotiatePlayerList() != null) {
                for (Player l_negotiatedPlayer : d_player.getD_negotiatePlayerList()) {
                    if (l_negotiatedPlayer.getD_ownedCountries().contains(l_countryTo)) {
+                        d_orderResponse.setD_isValid(false);
+                        d_orderResponse.setD_responseString("Opponent's Player is negotiated");
                        return false;
                    }
                }
@@ -66,6 +68,8 @@ public boolean executeOrder() {
                int l_fromArmies = l_countryfrom.getD_noOfArmies();
                //returns if given no. of armies are higher than country has
                if (l_fromArmies < d_noOfArmies) {
+                    d_orderResponse.setD_isValid(false);
+                    d_orderResponse.setD_responseString("given no. of armies are higher than country have");
                    return false;
                }
                //condition matches if both countries owned by same player and countryto is neighbour to countryfrom
@@ -92,6 +96,7 @@ public boolean executeOrder() {
                    d_gameData.getD_playerList().remove(l_playerFromIndex);
                    d_gameData.getD_playerList().add(l_playerFromIndex, d_player);

+                    d_orderResponse.setD_isValid(true);
                    return true;
                } else {
                    //checks for attack to intialize if country to is not owned by the same player
```

- User Interface –



- Removed unused variable –