SOEN 6441 Advanced Programming Practices

REFACTORING DOCUMENTATION

Group W10 – Build 3

Group members:

- Omnia Alam
- Yajing Liu
- Sherwyn Dsouza
- Darlene Nazareth
- Duy Thanh Phan
- Md Tazin Morshed Shad

Potential Refactoring Targets

The potential refactoring targets are listed below. We determine the targets based on new requirements of Build 3 and problematic issues that happened when developing Build 1 and 2.

1. Implement the Adapter pattern for the application to support different map formats.
2. Implement the Strategy pattern for players' strategy.
3. Improve exception handling for incorrect commands and illegal states in the application.
4. Improve naming conventions for classes, functions and variables.
5. Improve the project folder structure to support maintainability.
6. Add additional test cases for the existing code base.
7. Remove unused imports, functions and variables.
8. Improve Javadoc content.
9. Reorganize Constants to be separated by responsibilities.
10. Use modern, recommended Java syntax to replace some existing code snippets.
11. Named functions descriptively and to ensure Single Responsibility Principle.
12. Restructure gameplay flow for single player mode and tournament mode.
13. Refactor validation logic location.
14. Multi-threading support for multiple player mode.
15. Improve error messages and game instruction.

Actual Refactoring Targets

1. Implement the Adapter pattern for the application to support different map formats.

Reasons:

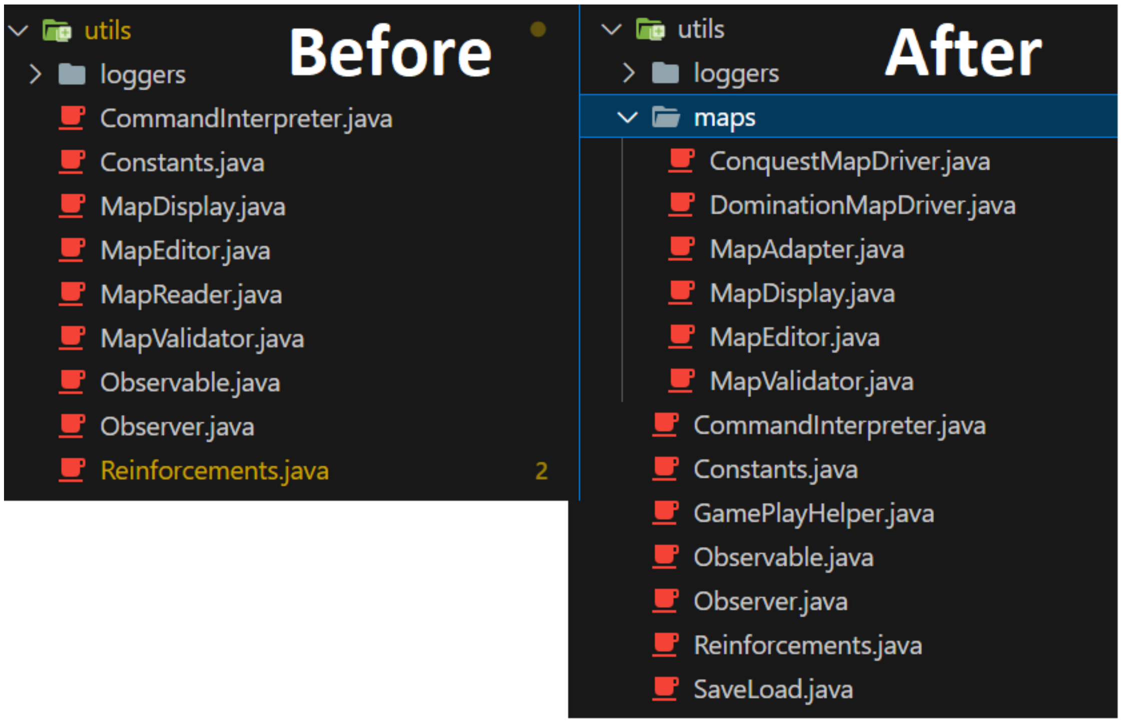
- The code at build 2 only supports Domination file format.
- Rewrite the code to support a new Conquest file format is going to take a major effort.
- In this situation, it is beneficial to write new adapters to transform the Conquest filemap to/from the Domination format.

Test commands:

- Load a Conquest format map file: `loadmap src/main/resources/maps/europe-conquest.map`
- Save map: `savemap newmap.map 1` - Domination (current) format or `savemap newmap.map 2` - Conquest format

Change summary:

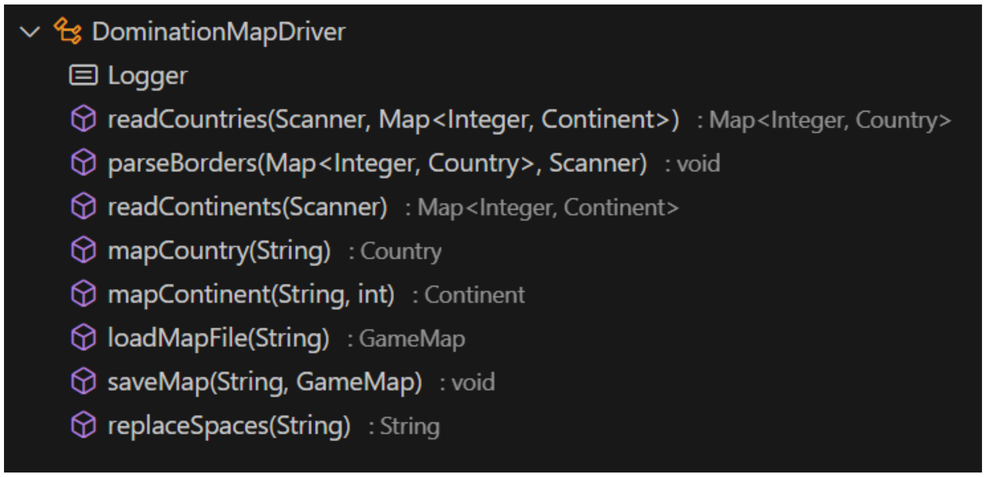
- `savemap` now accepts a 2nd argument for map format: `1` - Domination map format, `2` - Conquest map format; leave empty => domination map format.
- Consolidate map functions:
    - all map-related files are now stored in `utils/maps` folder. Test files are also moved accordingly.



```
Before
utils
  loggers
  CommandInterpreter.java
  Constants.java
  MapDisplay.java
  MapEditor.java
  MapReader.java
  MapValidator.java
  Observable.java
  Observer.java
  Reinforcements.java    2
```

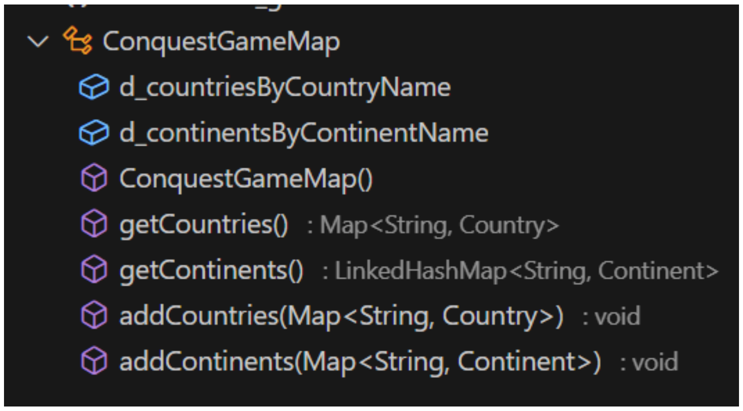
```
After
utils
  loggers
  maps
    ConquestMapDriver.java
    DominationMapDriver.java
    MapAdapter.java
    MapDisplay.java
    MapEditor.java
    MapValidator.java
  CommandInterpreter.java
  Constants.java
  GamePlayHelper.java
  Observable.java
  Observer.java
  Reinforcements.java
  SaveLoad.java
```

- renamed the existing `MapReader` class to `DominationMapDriver`. A (Domination|Conquest)`MapDriver` instance now contains methods for both reading and saving the map. `saveMap` method was refactored to move from the specific GameMap class to the corresponding *`MapDriver` class.



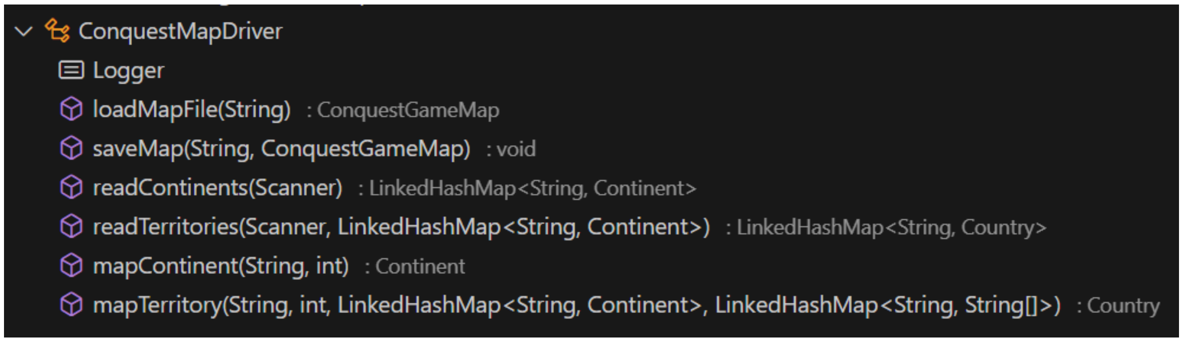
```
DominationMapDriver
    Logger
    readCountries(Scanner, Map<Integer, Continent>)  : Map<Integer, Country>
    parseBorders(Map<Integer, Country>, Scanner)  : void
    readContinents(Scanner)  : Map<Integer, Continent>
    mapCountry(String)  : Country
    mapContinent(String, int)  : Continent
    loadMapFile(String)  : GameMap
    saveMap(String, GameMap)  : void
    replaceSpaces(String)  : String
```

- added `ConquestGameMap` to support storing game map data specific to Conquest map file beside the existing `GameMap` class that supports Domination map.



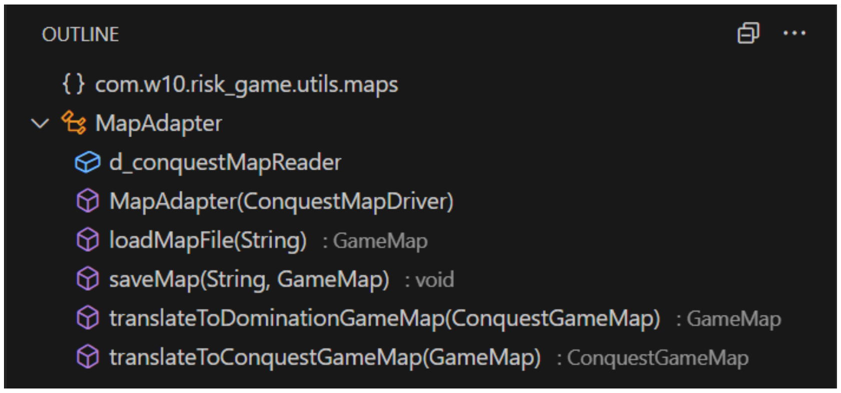
```
ConquestGameMap
    d_countriesByCountryName
    d_continentsByContinentName
    ConquestGameMap()
    getCountries()  : Map<String, Country>
    getContinents()  : LinkedHashMap<String, Continent>
    addCountries(Map<String, Country>)  : void
    addContinents(Map<String, Continent>)  : void
```

- added `ConquestMapDriver` to make changes to the instances of `ConquestGameMap`



```
ConquestMapDriver
    Logger
    loadMapFile(String)  : ConquestGameMap
    saveMap(String, ConquestGameMap)  : void
    readContinents(Scanner)  : LinkedHashMap<String, Continent>
    readTerritories(Scanner, LinkedHashMap<String, Continent>)  : LinkedHashMap<String, Country>
    mapContinent(String, int)  : Continent
    mapTerritory(String, int, LinkedHashMap<String, Continent>, LinkedHashMap<String, String[]>)  : Country
```

- added `MapAdapter` to handle transformation between `GameMap` and `ConquestGameMap`

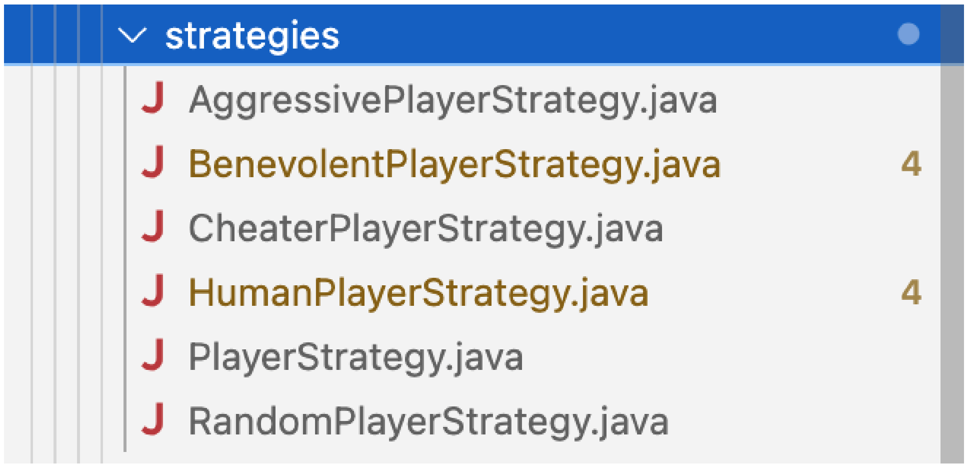

OUTLINE

```
{ } com.w10.risk_game.utils.maps
MapAdapter
    d_conquestMapReader
    MapAdapter(ConquestMapDriver)
    loadMapFile(String) : GameMap
    saveMap(String, GameMap) : void
    translateToDominationGameMap(ConquestGameMap) : GameMap
    translateToConquestGameMap(GameMap) : ConquestGameMap
```

2. Use Strategy Pattern to implements different types of Player's strategies:

Reasons:

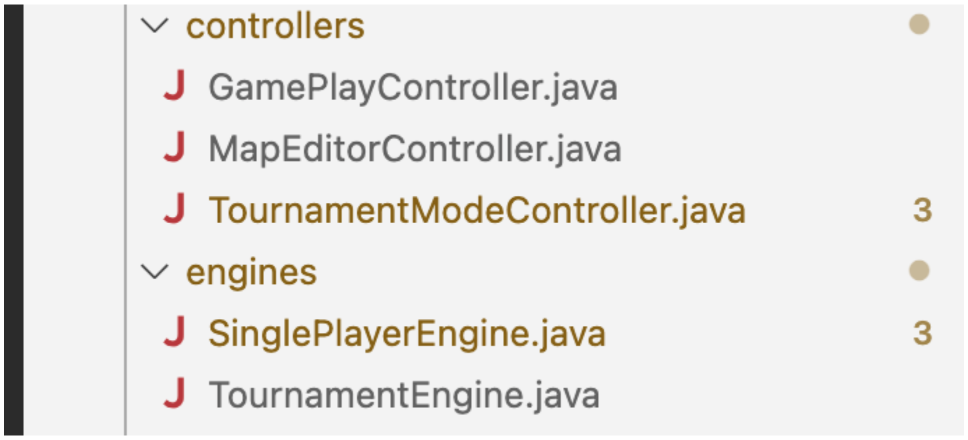
- Since this build requires us to develop different types of strategies for the players and aslo to develop an automated tournament based on different player strategies and the best way to accomplish that by using Strategy pattern
- The main player class remain same but we have added strategy feature to the player.
- When the palyer is created , if we define which strategy the will be playing as then the player strategy will be automatically assigned to the player or else we can still play as a Human player.
- It helps us for future extension, as we can add as many strategies we want without breaking the existing code.


```
strategies
    AggressivePlayerStrategy.java
    BenevolentPlayerStrategy.java      4
    CheaterPlayerStrategy.java
    HumanPlayerStrategy.java           4
    PlayerStrategy.java
    RandomPlayerStrategy.java
```

3. Restructure gameplay flow for single player mode and tournament mode

Reasons:

- In Build 3 we have 2 game play modes so that is why we structured our GameEngine into two to support different modes that are SinglePlayerEngine and TournamentEngine.
- We have Single player mode and Tournament mode. In the single player mode there will be human interaction and in the tournament mode the orders will be automated.
- In order to support that we split our GamePlayEngine into 2 Engines but the logic of the Game Play has remained same.
- Also to support the tournament commands we have added an extra controller as well.

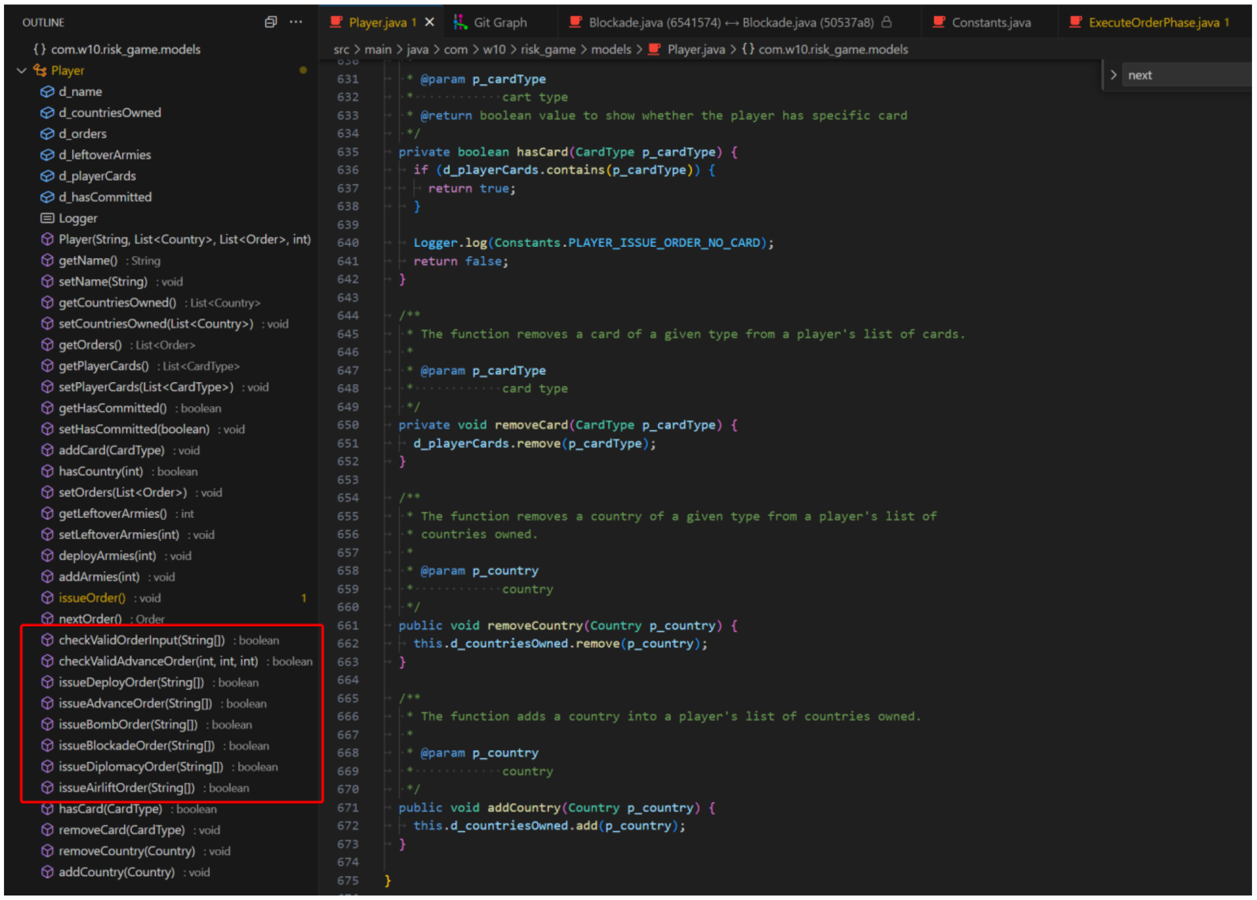


```
controllers
    GamePlayController.java
    MapEditorController.java
    TournamentModeController.java    3
engines
    SinglePlayerEngine.java          3
    TournamentEngine.java
```

4. Refactor validation and issue order logic location

Reasons:

- In build 2, we put the logic for validating and issuing different kinds of orders in the same Player.java file.
- This violates the separation of concerns and make the file too big for maintenance (676 lines)
- The solution that we chose is to move them to the dedicated Command class, so the logic is encapsulated and be relevant to the business domain.



OUTLINE

Player.java 1 · Git Graph · Blockade.java (6541574) ↔ Blockade.java (50537a8) · Constants.java · ExecuteOrderPhase.java 1

src > main > java > com > w10 > risk_game > models > Player.java > {} com.w10.risk_game.models

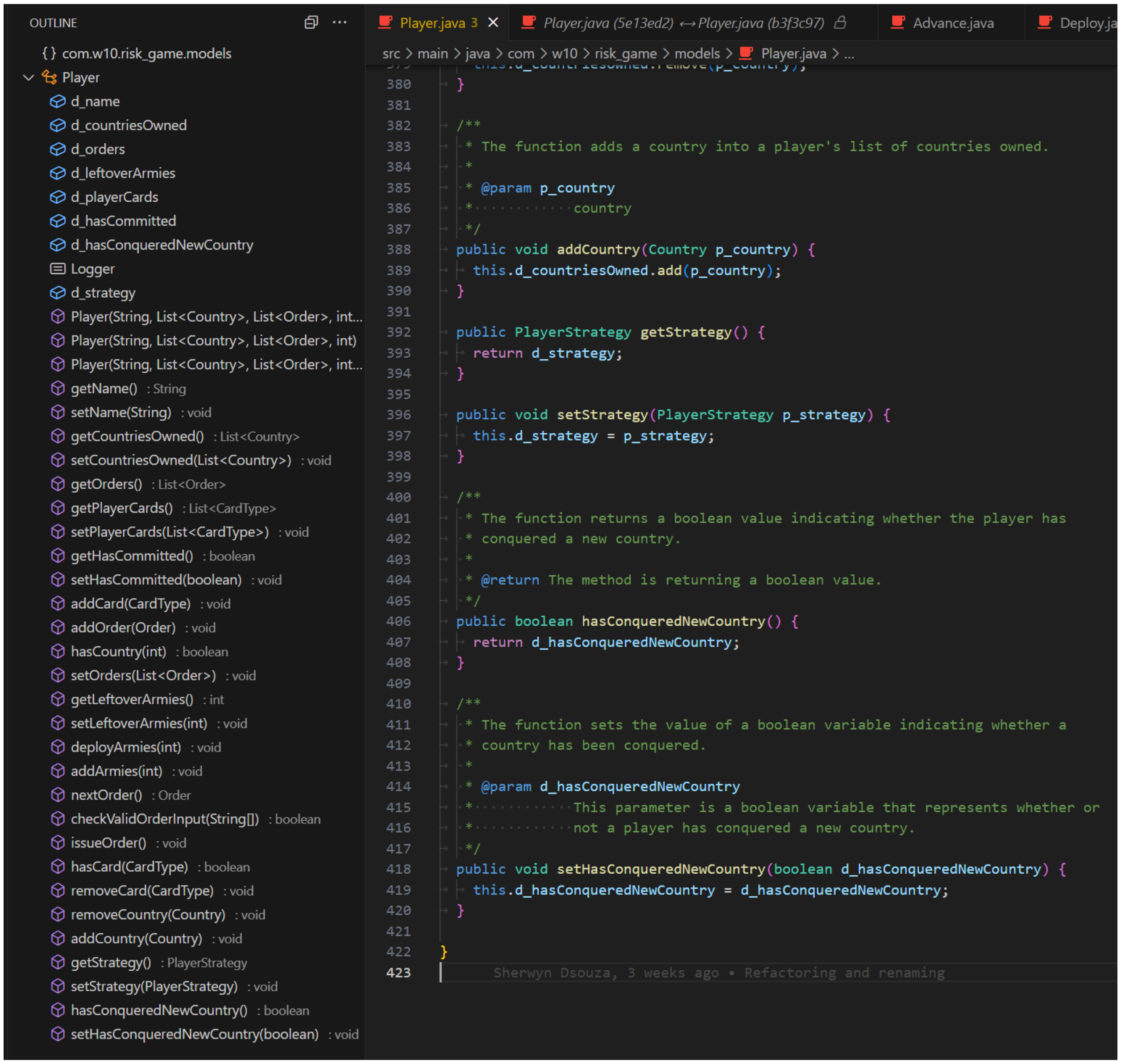
```
{} com.w10.risk_game.models
Player
  d_name
  d_countriesOwned
  d_orders
  d_leftoverArmies
  d_playerCards
  d_hasCommitted
  Logger
  Player(String, List<Country>, List<Order>, int)
  getName()  : String
  setName(String)  : void
  getCountriesOwned()  : List<Country>
  setCountriesOwned(List<Country>)  : void
  getOrders()  : List<Order>
  getPlayerCards()  : List<CardType>
  setPlayerCards(List<CardType>)  : void
  getHasCommitted()  : boolean
  setHasCommitted(boolean)  : void
  addCard(CardType)  : void
  hasCountry(int)  : boolean
  setOrders(List<Order>)  : void
  getLeftoverArmies()  : int
  setLeftoverArmies(int)  : void
  deployArmies(int)  : void
  addArmies(int)  : void
  issueOrder()  : void                                   1
  nextOrder()  : Order
  checkValidOrderInput(String[])  : boolean
  checkValidAdvanceOrder(int, int, int)  : boolean
  issueDeployOrder(String[])  : boolean
  issueAdvanceOrder(String[])  : boolean
  issueBombOrder(String[])  : boolean
  issueBlockadeOrder(String[])  : boolean
  issueDiplomacyOrder(String[])  : boolean
  issueAirliftOrder(String[])  : boolean
  hasCard(CardType)  : boolean
  removeCard(CardType)  : void
  removeCountry(Country)  : void
  addCountry(Country)  : void
```

```java
	 * @param p_cardType
	 *            cart type
	 * @return boolean value to show whether the player has specific card
	 */
	private boolean hasCard(CardType p_cardType) {
		if (d_playerCards.contains(p_cardType)) {
			return true;
		}

		Logger.log(Constants.PLAYER_ISSUE_ORDER_NO_CARD);
		return false;
	}

	/**
	 * The function removes a card of a given type from a player's list of cards.
	 *
	 * @param p_cardType
	 *            card type
	 */
	private void removeCard(CardType p_cardType) {
		d_playerCards.remove(p_cardType);
	}

	/**
	 * The function removes a country of a given type from a player's list of
	 * countries owned.
	 *
	 * @param p_country
	 *            country
	 */
	public void removeCountry(Country p_country) {
		this.d_countriesOwned.remove(p_country);
	}

	/**
	 * The function adds a country into a player's list of countries owned.
	 *
	 * @param p_country
	 *            country
	 */
	public void addCountry(Country p_country) {
		this.d_countriesOwned.add(p_country);
	}

}
```

Figure 1 Code outline of Player.java in Build 2

Change summary:

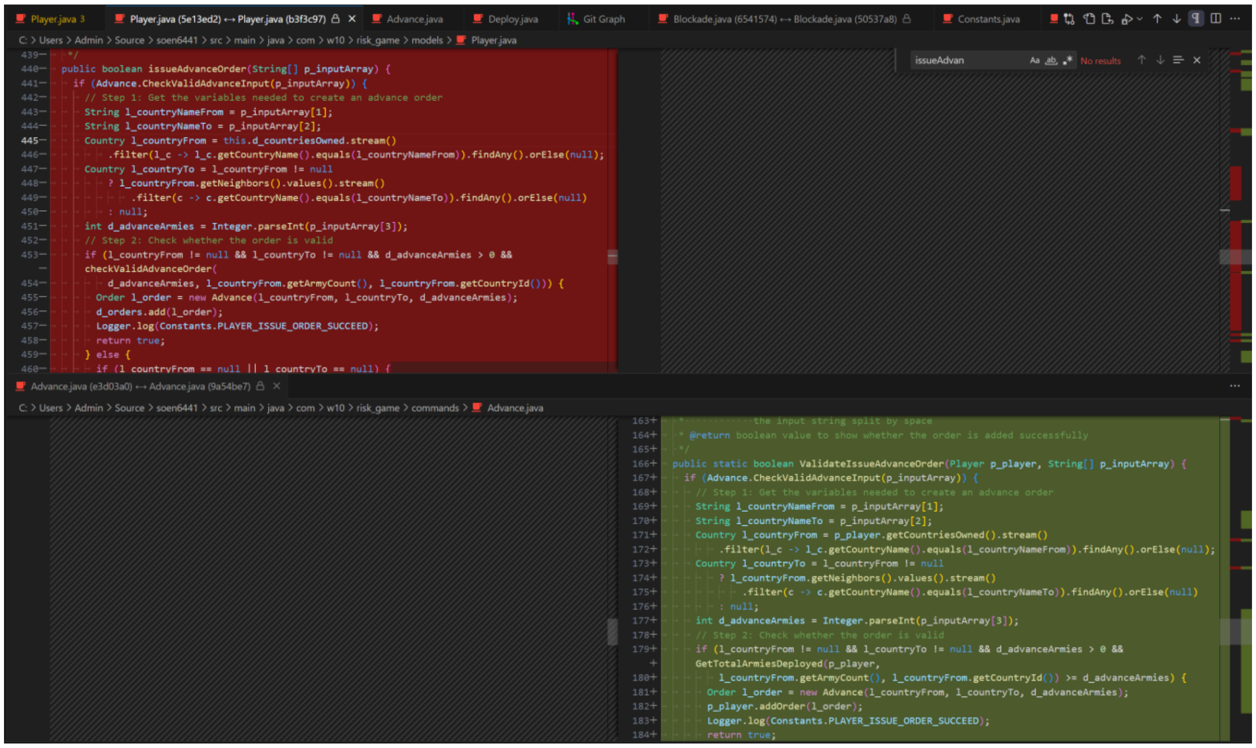
- We move the functions for issuing orders to be static function of the dedicated Command classes
  + Player.java (issueDeployOrder) -> commands/Deploy.java (ValidateIssueDeployOrder)
  + Player.java (issueAdvanceOrder) -> commands/Advance.java (ValidateIssueAdvanceOrder)
  + Player.java (issueBombOrder) -> commands/Bomb.java (ValidateIssueBombOrder)
  + Player.java (issueBlockadeOrder) -> commands/Blockade.java (ValidateIssueBlockadeOrder)
  + Player.java (issueDiplomacyOrder) -> commands/Negotiate.java (ValidateIssueDiplomacyOrder)
  + Player.java (issueAirliftOrder) -> commands/Airlift.java (ValidateIssueAirliftOrder)


OUTLINE
Player.java 3
Player.java (5e13ed2) ↔ Player.java (b3f3c97)
Advance.java
Deploy.ja
src > main > java > com > w10 > risk_game > models > Player.java > ...

```
{ } com.w10.risk_game.models
  Player
    d_name
    d_countriesOwned
    d_orders
    d_leftoverArmies
    d_playerCards
    d_hasCommitted
    d_hasConqueredNewCountry
    Logger
    d_strategy
    Player(String, List<Country>, List<Order>, int...
    Player(String, List<Country>, List<Order>, int)
    Player(String, List<Country>, List<Order>, int...
    getName()  : String
    setName(String)  : void
    getCountriesOwned()  : List<Country>
    setCountriesOwned(List<Country>)  : void
    getOrders()  : List<Order>
    getPlayerCards()  : List<CardType>
    setPlayerCards(List<CardType>)  : void
    getHasCommitted()  : boolean
    setHasCommitted(boolean)  : void
    addCard(CardType)  : void
    addOrder(Order)  : void
    hasCountry(int)  : boolean
    setOrders(List<Order>)  : void
    getLeftoverArmies()  : int
    setLeftoverArmies(int)  : void
    deployArmies(int)  : void
    addArmies(int)  : void
    nextOrder()  : Order
    checkValidOrderInput(String[])  : boolean
    issueOrder()  : void
    hasCard(CardType)  : boolean
    removeCard(CardType)  : void
    removeCountry(Country)  : void
    addCountry(Country)  : void
    getStrategy()  : PlayerStrategy
    setStrategy(PlayerStrategy)  : void
    hasConqueredNewCountry()  : boolean
    setHasConqueredNewCountry(boolean)  : void
```

```java
        this.d_countriesOwned.remove(p_country);
    }

    /**
     * The function adds a country into a player's list of countries owned.
     *
     * @param p_country
     *            country
     */
    public void addCountry(Country p_country) {
        this.d_countriesOwned.add(p_country);
    }

    public PlayerStrategy getStrategy() {
        return d_strategy;
    }

    public void setStrategy(PlayerStrategy p_strategy) {
        this.d_strategy = p_strategy;
    }

    /**
     * The function returns a boolean value indicating whether the player has
     * conquered a new country.
     *
     * @return The method is returning a boolean value.
     */
    public boolean hasConqueredNewCountry() {
        return d_hasConqueredNewCountry;
    }

    /**
     * The function sets the value of a boolean variable indicating whether a
     * country has been conquered.
     *
     * @param d_hasConqueredNewCountry
     *            This parameter is a boolean variable that represents whether or
     *            not a player has conquered a new country.
     */
    public void setHasConqueredNewCountry(boolean d_hasConqueredNewCountry) {
        this.d_hasConqueredNewCountry = d_hasConqueredNewCountry;
    }

}
        Sherwyn Dsouza, 3 weeks ago • Refactoring and renaming
```

Figure 2 Code layout of Player.java in Build 3


Player.java 3
Player.java (5e13ed2) ↔ Player.java (b3f3c97)
Advance.java
Deploy.java
Git Graph
Blockade.java (6541574) ↔ Blockade.java (50537a8)
Constants.java
C: > Users > Admin > Source > soen6441 > src > main > java > com > w10 > risk_game > models > Player.java
issueAdvan
No results
Advance.java (e3d03a0) ↔ Advance.java (9a54be7)
C: > Users > Admin > Source > soen6441 > src > main > java > com > w10 > risk_game > commands > Advance.java

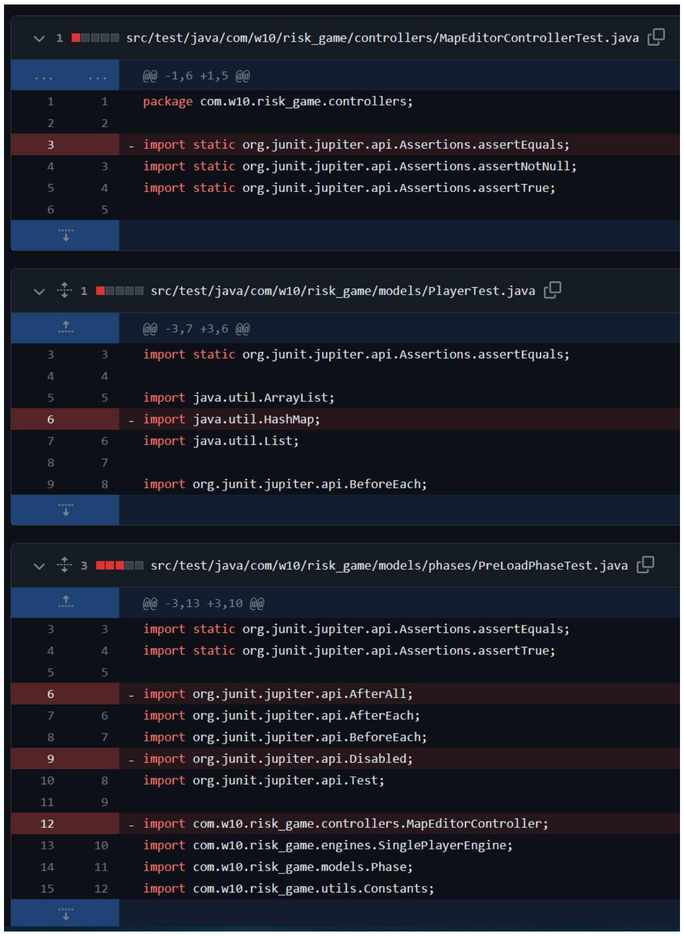
```java
 */
public boolean issueAdvanceOrder(String[] p_inputArray) {
  if (Advance.CheckValidAdvanceInput(p_inputArray)) {
    // Step 1: Get the variables needed to create an advance order
    String l_countryNameFrom = p_inputArray[1];
    String l_countryNameTo = p_inputArray[2];
    Country l_countryFrom = this.d_countriesOwned.stream()
        .filter(l_c -> l_c.getCountryName().equals(l_countryNameFrom)).findAny().orElse(null);
    Country l_countryTo = l_countryFrom != null
        ? l_countryFrom.getNeighbors().values().stream()
            .filter(c -> c.getCountryName().equals(l_countryNameTo)).findAny().orElse(null)
        : null;
    int d_advanceArmies = Integer.parseInt(p_inputArray[3]);
    // Step 2: Check whether the order is valid
    if (l_countryFrom != null && l_countryTo != null && d_advanceArmies > 0 &&
    checkValidAdvanceOrder(
        d_advanceArmies, l_countryFrom.getArmyCount(), l_countryFrom.getCountryId())) {
      Order l_order = new Advance(l_countryFrom, l_countryTo, d_advanceArmies);
      d_orders.add(l_order);
      Logger.log(Constants.PLAYER_ISSUE_ORDER_SUCCEED);
      return true;
    } else {
      if (l_countryFrom == null || l_countryTo == null) {
```

```java
 *            the input string split by space
 * @return boolean value to show whether the order is added successfully
 */
public static boolean ValidateIssueAdvanceOrder(Player p_player, String[] p_inputArray) {
  if (Advance.CheckValidAdvanceInput(p_inputArray)) {
    // Step 1: Get the variables needed to create an advance order
    String l_countryNameFrom = p_inputArray[1];
    String l_countryNameTo = p_inputArray[2];
    Country l_countryFrom = p_player.getCountriesOwned().stream()
        .filter(l_c -> l_c.getCountryName().equals(l_countryNameFrom)).findAny().orElse(null);
    Country l_countryTo = l_countryFrom != null
        ? l_countryFrom.getNeighbors().values().stream()
            .filter(c -> c.getCountryName().equals(l_countryNameTo)).findAny().orElse(null)
        : null;
    int d_advanceArmies = Integer.parseInt(p_inputArray[3]);
    // Step 2: Check whether the order is valid
    if (l_countryFrom != null && l_countryTo != null && d_advanceArmies > 0 &&
    GetTotalArmiesDeployed(p_player,
        l_countryFrom.getArmyCount(), l_countryFrom.getCountryId()) >= d_advanceArmies) {
      Order l_order = new Advance(l_countryFrom, l_countryTo, d_advanceArmies);
      p_player.addOrder(l_order);
      Logger.log(Constants.PLAYER_ISSUE_ORDER_SUCCEED);
      return true;
```

Figure 3 Example of code refactoring for validating and issuing order

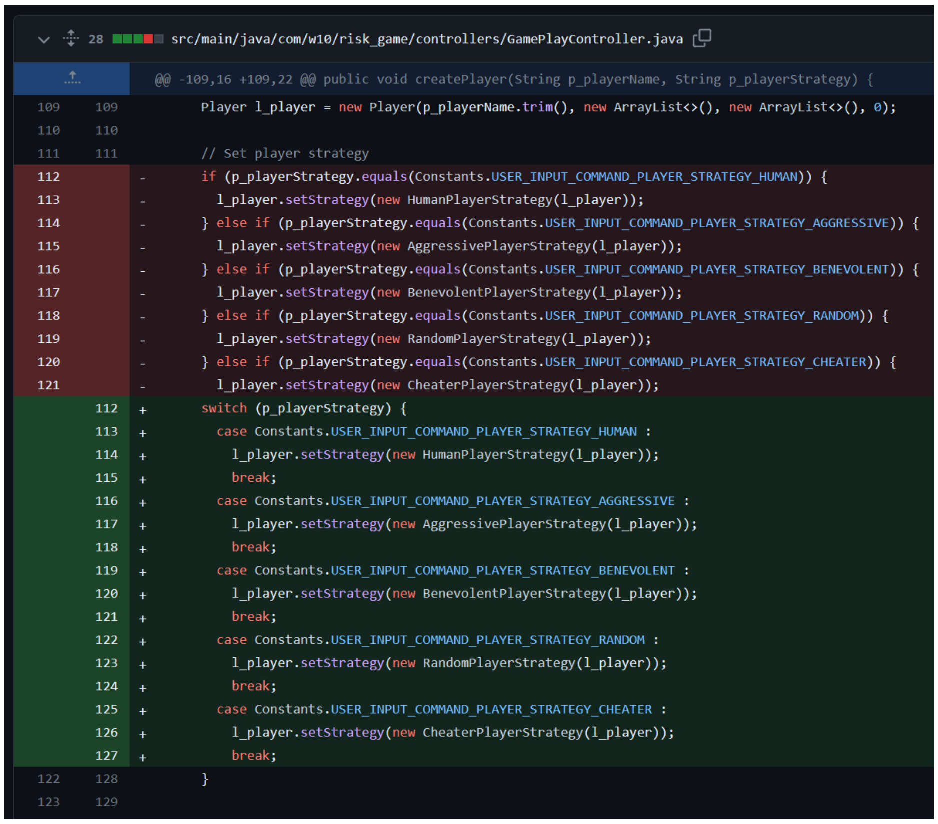
5. Use modern, recommended Java syntax to replace some existing code snippets.

Remove unused imports



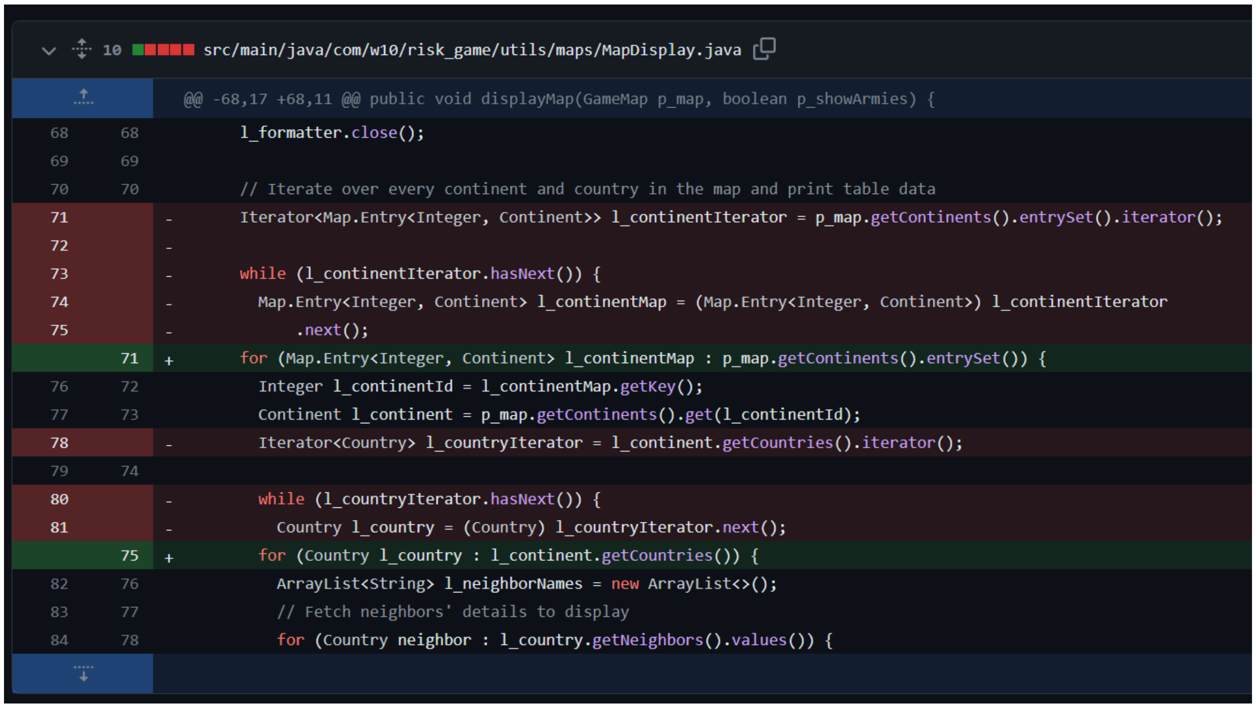
```diff
1 src/test/java/com/w10/risk_game/controllers/MapEditorControllerTest.java
@@ -1,6 +1,5 @@
 package com.w10.risk_game.controllers;
 
-import static org.junit.jupiter.api.Assertions.assertEquals;
 import static org.junit.jupiter.api.Assertions.assertNotNull;
 import static org.junit.jupiter.api.Assertions.assertTrue;
 

1 src/test/java/com/w10/risk_game/models/PlayerTest.java
@@ -3,7 +3,6 @@
 import static org.junit.jupiter.api.Assertions.assertEquals;
 
 import java.util.ArrayList;
-import java.util.HashMap;
 import java.util.List;
 
 import org.junit.jupiter.api.BeforeEach;

3 src/test/java/com/w10/risk_game/models/phases/PreLoadPhaseTest.java
@@ -3,13 +3,10 @@
 import static org.junit.jupiter.api.Assertions.assertEquals;
 import static org.junit.jupiter.api.Assertions.assertTrue;
 
-import org.junit.jupiter.api.AfterAll;
 import org.junit.jupiter.api.AfterEach;
 import org.junit.jupiter.api.BeforeEach;
-import org.junit.jupiter.api.Disabled;
 import org.junit.jupiter.api.Test;
 
-import com.w10.risk_game.controllers.MapEditorController;
 import com.w10.risk_game.engines.SinglePlayerEngine;
 import com.w10.risk_game.models.Phase;
 import com.w10.risk_game.utils.Constants;
```

Change if else to switch case where necessary



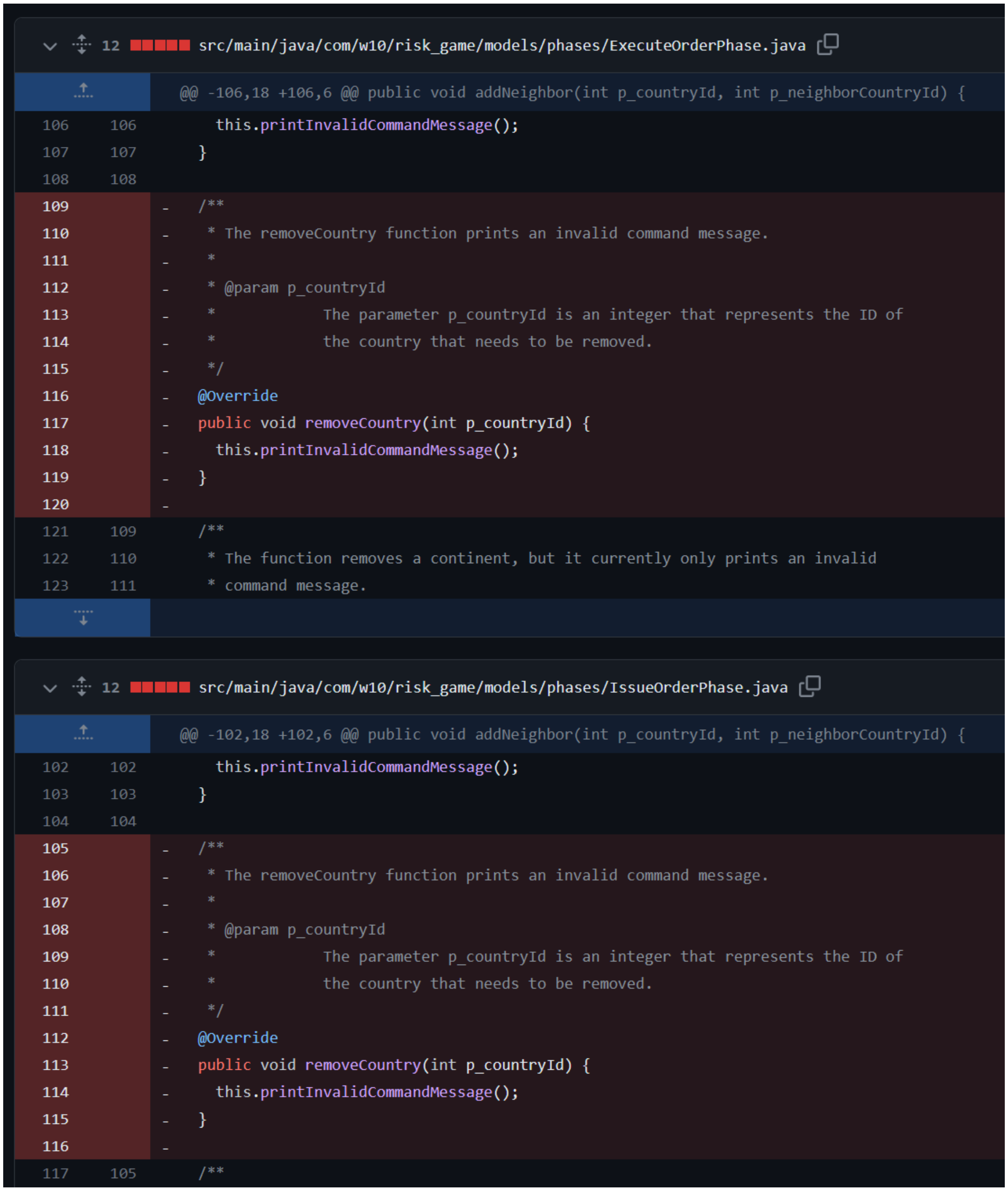
```diff
28 src/main/java/com/w10/risk_game/controllers/GamePlayController.java
@@ -109,16 +109,22 @@ public void createPlayer(String p_playerName, String p_playerStrategy) {
         Player l_player = new Player(p_playerName.trim(), new ArrayList<>(), new ArrayList<>(), 0);

         // Set player strategy
-        if (p_playerStrategy.equals(Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_HUMAN)) {
-          l_player.setStrategy(new HumanPlayerStrategy(l_player));
-        } else if (p_playerStrategy.equals(Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_AGGRESSIVE)) {
-          l_player.setStrategy(new AggressivePlayerStrategy(l_player));
-        } else if (p_playerStrategy.equals(Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_BENEVOLENT)) {
-          l_player.setStrategy(new BenevolentPlayerStrategy(l_player));
-        } else if (p_playerStrategy.equals(Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_RANDOM)) {
-          l_player.setStrategy(new RandomPlayerStrategy(l_player));
-        } else if (p_playerStrategy.equals(Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_CHEATER)) {
-          l_player.setStrategy(new CheaterPlayerStrategy(l_player));
+        switch (p_playerStrategy) {
+          case Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_HUMAN :
+            l_player.setStrategy(new HumanPlayerStrategy(l_player));
+            break;
+          case Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_AGGRESSIVE :
+            l_player.setStrategy(new AggressivePlayerStrategy(l_player));
+            break;
+          case Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_BENEVOLENT :
+            l_player.setStrategy(new BenevolentPlayerStrategy(l_player));
+            break;
+          case Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_RANDOM :
+            l_player.setStrategy(new RandomPlayerStrategy(l_player));
+            break;
+          case Constants.USER_INPUT_COMMAND_PLAYER_STRATEGY_CHEATER :
+            l_player.setStrategy(new CheaterPlayerStrategy(l_player));
+            break;
         }

```

Use enhanced for loop



```diff
10 src/main/java/com/w10/risk_game/utils/maps/MapDisplay.java
@@ -68,17 +68,11 @@ public void displayMap(GameMap p_map, boolean p_showArmies) {
         l_formatter.close();

         // Iterate over every continent and country in the map and print table data
-        Iterator<Map.Entry<Integer, Continent>> l_continentIterator = p_map.getContinents().entrySet().iterator();
-
-        while (l_continentIterator.hasNext()) {
-          Map.Entry<Integer, Continent> l_continentMap = (Map.Entry<Integer, Continent>) l_continentIterator
-              .next();
+        for (Map.Entry<Integer, Continent> l_continentMap : p_map.getContinents().entrySet()) {
           Integer l_continentId = l_continentMap.getKey();
           Continent l_continent = p_map.getContinents().get(l_continentId);
-          Iterator<Country> l_countryIterator = l_continent.getCountries().iterator();

-          while (l_countryIterator.hasNext()) {
-            Country l_country = (Country) l_countryIterator.next();
+          for (Country l_country : l_continent.getCountries()) {
             ArrayList<String> l_neighborNames = new ArrayList<>();
             // Fetch neighbors' details to display
             for (Country neighbor : l_country.getNeighbors().values()) {
```

Remove unnecessary overridden methods



12 src/main/java/com/w10/risk_game/models/phases/ExecuteOrderPhase.java

```
@@ -106,18 +106,6 @@ public void addNeighbor(int p_countryId, int p_neighborCountryId) {
        this.printInvalidCommandMessage();
      }

-     /**
-      * The removeCountry function prints an invalid command message.
-      *
-      * @param p_countryId
-      *             The parameter p_countryId is an integer that represents the ID of
-      *             the country that needs to be removed.
-      */
-     @Override
-     public void removeCountry(int p_countryId) {
-       this.printInvalidCommandMessage();
-     }
-
      /**
       * The function removes a continent, but it currently only prints an invalid
       * command message.
```

12 src/main/java/com/w10/risk_game/models/phases/IssueOrderPhase.java

```
@@ -102,18 +102,6 @@ public void addNeighbor(int p_countryId, int p_neighborCountryId) {
        this.printInvalidCommandMessage();
      }

-     /**
-      * The removeCountry function prints an invalid command message.
-      *
-      * @param p_countryId
-      *             The parameter p_countryId is an integer that represents the ID of
-      *             the country that needs to be removed.
-      */
-     @Override
-     public void removeCountry(int p_countryId) {
-       this.printInvalidCommandMessage();
-     }
-
      /**
```