Group U5: REFACTORING BUILD 3
COURSE: SOEN 6441
INSTRUCTOR: AMIN RANJ BAR

TEAM MEMBERS:

- Abhishek Handa (40231719)
- Rajat Sharma (40196467)
- Harman Singh Jolly (40204947)
- Amanpreet Singh (40221947)
- Anurag Teckchandani (40263724)

Potential Refactoring Targets:

The following refactoring target list has been taken into consideration from the requirements gathered from in-consistentencies and pain points of build2 and during the complete development of build 3.

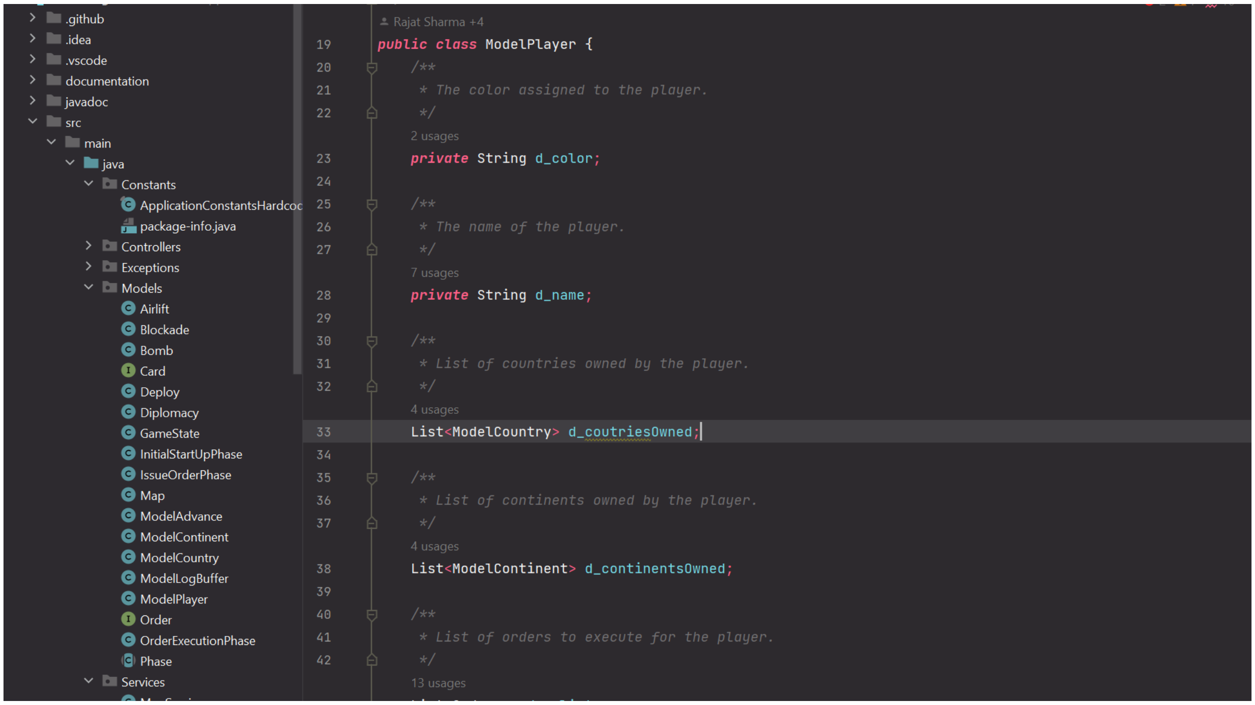
1) Implementation of Observer pattern for console logs.
2) Refactoring Adapter pattern from the loading and saving of Domination and Conquest types of maps
3) Refactoring Strategy pattern to player behavioral strategies.
4) Better display of information on the console.
5) Refactoring error handling via observer pattern.
6) Modularized and Completely separated observer to view directory.
7) Game Refactoring with single & tournament mode.
8) Command pattern Validation.
9) Refactoring and correction according to coding convention.

10) Additional test cases implementation.
11) Added country neutral list.
12) Save Game and Load Game addition.
13) Neutral country list addition.
14) Refactoring with Save Game and Load Game.
15) Javadoc addition to private data members

Actual Refactoring Targets:

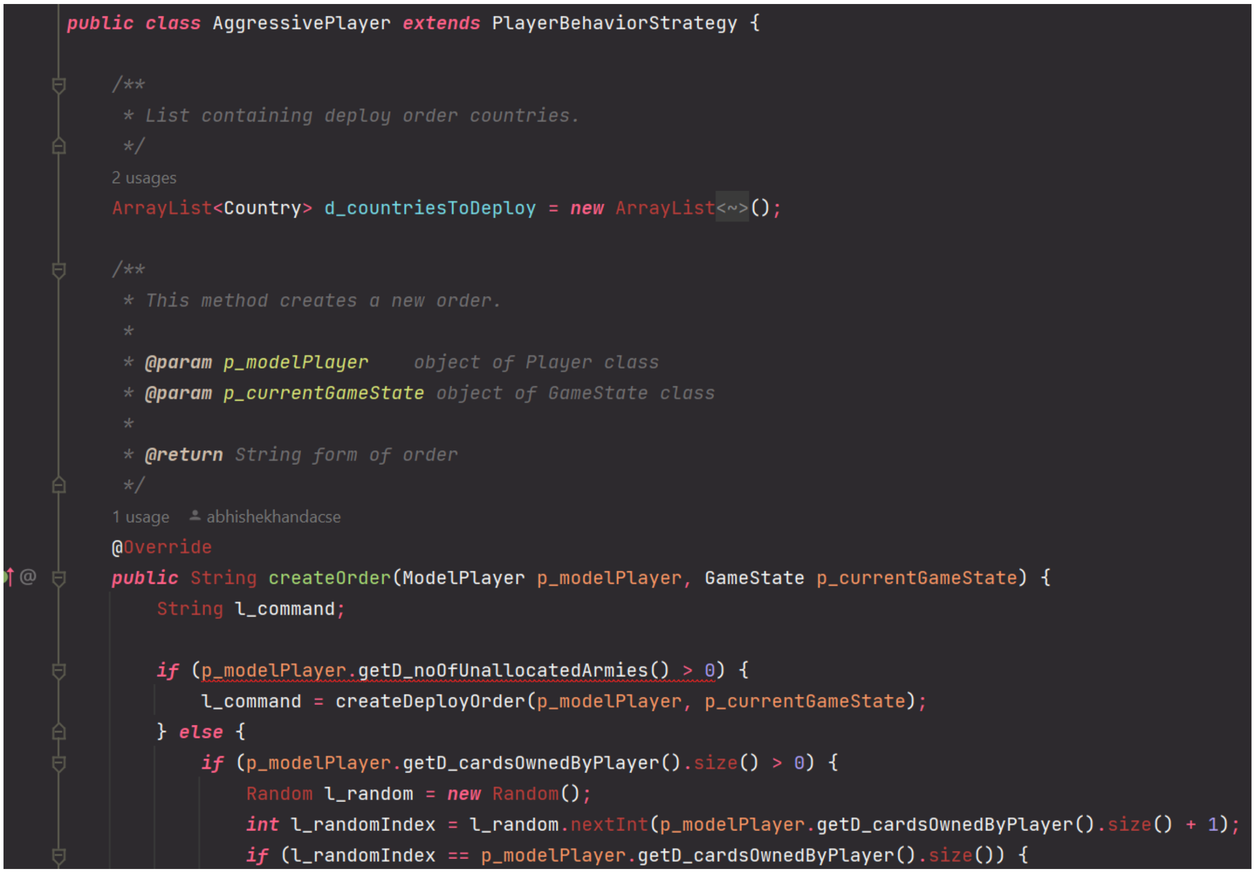
1) Refactor the Player's issueOrder() method to use Strategy pattern:
With respect to the requirements of build3, we have refactored the issueOrder() method of the class player and every player strategy.

Before Refactoring



```
Rajat Sharma +4
public class ModelPlayer {
    /**
     * The color assigned to the player.
     */
    2 usages
    private String d_color;

    /**
     * The name of the player.
     */
    7 usages
    private String d_name;

    /**
     * List of countries owned by the player.
     */
    4 usages
    List<ModelCountry> d_coutriesOwned;

    /**
     * List of continents owned by the player.
     */
    4 usages
    List<ModelContinent> d_continentsOwned;

    /**
     * List of orders to execute for the player.
     */
    13 usages
```

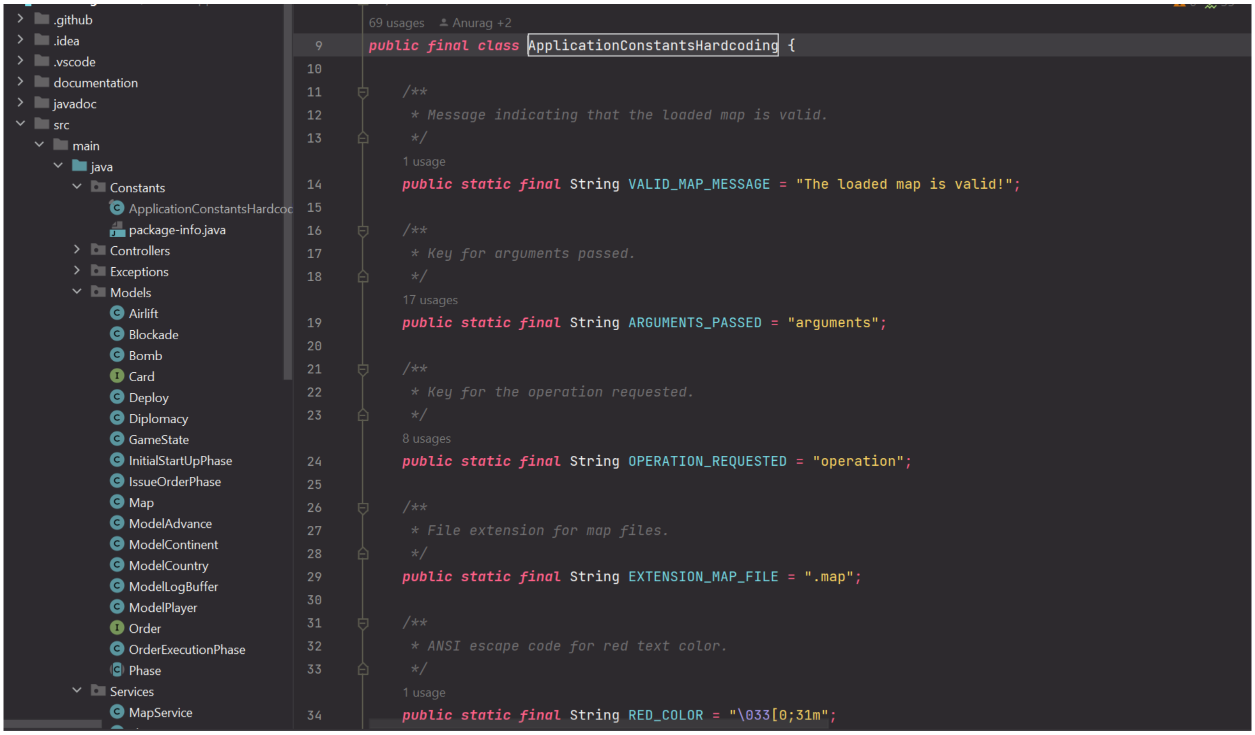
After Refactoring



```java
public class AggressivePlayer extends PlayerBehaviorStrategy {

    /**
     * List containing deploy order countries.
     */
    2 usages
    ArrayList<Country> d_countriesToDeploy = new ArrayList<~>();

    /**
     * This method creates a new order.
     *
     * @param p_modelPlayer     object of Player class
     * @param p_currentGameState object of GameState class
     *
     * @return String form of order
     */
    1 usage  abhishekhandacse
    @Override
    public String createOrder(ModelPlayer p_modelPlayer, GameState p_currentGameState) {
        String l_command;

        if (p_modelPlayer.getD_noOfUnallocatedArmies() > 0) {
            l_command = createDeployOrder(p_modelPlayer, p_currentGameState);
        } else {
            if (p_modelPlayer.getD_cardsOwnedByPlayer().size() > 0) {
                Random l_random = new Random();
                int l_randomIndex = l_random.nextInt(p_modelPlayer.getD_cardsOwnedByPlayer().size() + 1);
                if (l_randomIndex == p_modelPlayer.getD_cardsOwnedByPlayer().size()) {
```

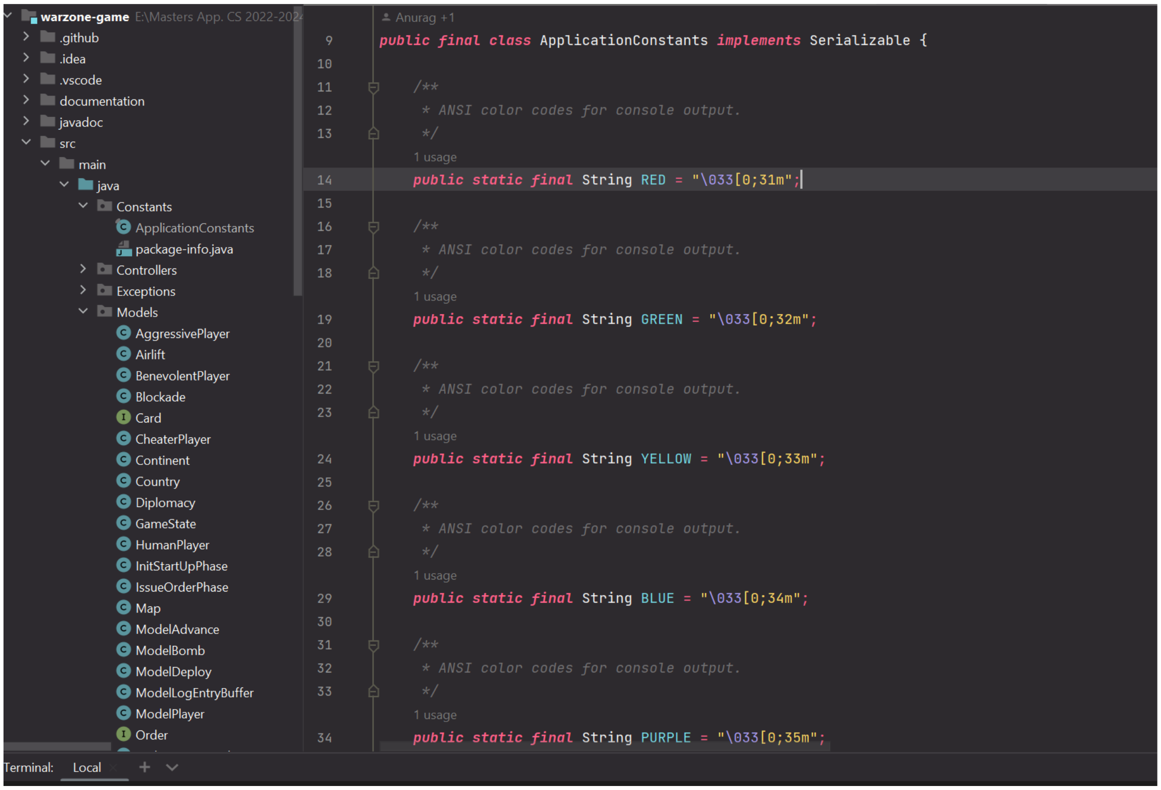
2) New ApplicationConstants file created and ApplicationConstantsHardcoding file removed.

Before Refactor



```java
public final class ApplicationConstantsHardcoding {

    /**
     * Message indicating that the loaded map is valid.
     */
    public static final String VALID_MAP_MESSAGE = "The loaded map is valid!";

    /**
     * Key for arguments passed.
     */
    public static final String ARGUMENTS_PASSED = "arguments";

    /**
     * Key for the operation requested.
     */
    public static final String OPERATION_REQUESTED = "operation";

    /**
     * File extension for map files.
     */
    public static final String EXTENSION_MAP_FILE = ".map";

    /**
     * ANSI escape code for red text color.
     */
    public static final String RED_COLOR = "\033[0;31m";
```

After Refactor



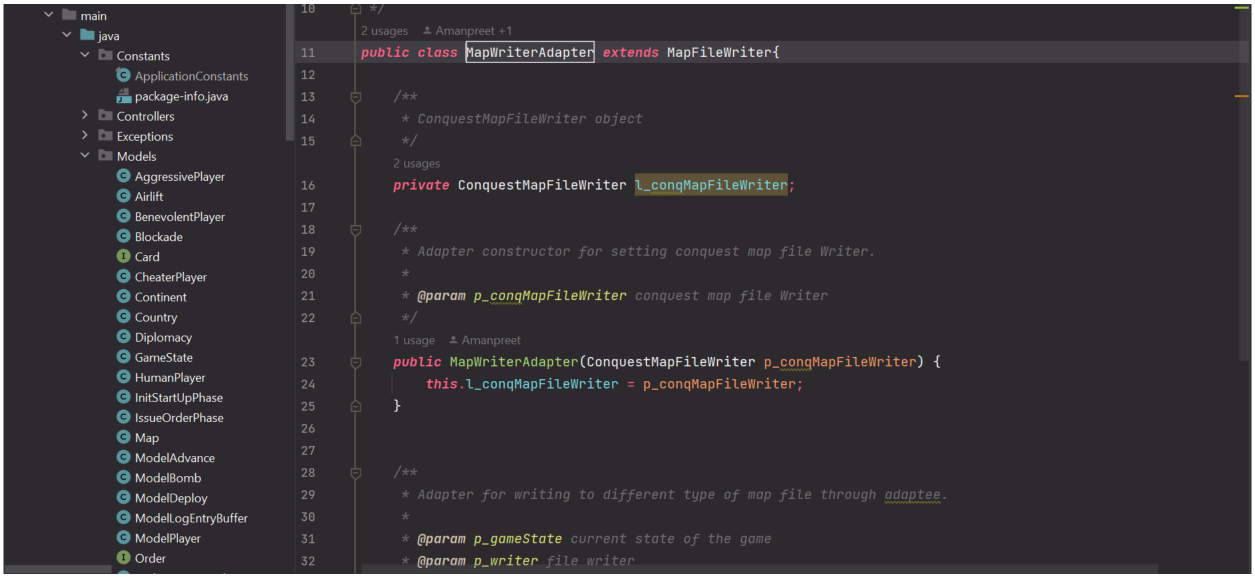
```
public final class ApplicationConstants implements Serializable {

    /**
     * ANSI color codes for console output.
     */
    public static final String RED = "\033[0;31m";

    /**
     * ANSI color codes for console output.
     */
    public static final String GREEN = "\033[0;32m";

    /**
     * ANSI color codes for console output.
     */
    public static final String YELLOW = "\033[0;33m";

    /**
     * ANSI color codes for console output.
     */
    public static final String BLUE = "\033[0;34m";

    /**
     * ANSI color codes for console output.
     */
    public static final String PURPLE = "\033[0;35m";
```

3) Applying adapter pattern for loading and saving domination files in build3.

Before Refactor:

Adapter was not present in build2

After Refactor:

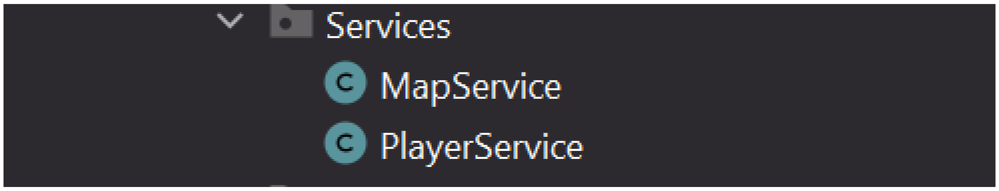


```
main
  java
    Constants
      ApplicationConstants
      package-info.java
    Controllers
    Exceptions
    Models
      AggressivePlayer
      Airlift
      BenevolentPlayer
      Blockade
      Card
      CheaterPlayer
      Continent
      Country
      Diplomacy
      GameState
      HumanPlayer
      InitStartUpPhase
      IssueOrderPhase
      Map
      ModelAdvance
      ModelBomb
      ModelDeploy
      ModelLogEntryBuffer
      ModelPlayer
      Order
```

```java
*/
2 usages   Amanpreet +1
public class MapWriterAdapter extends MapFileWriter{

    /**
     * ConquestMapFileWriter object
     */
    2 usages
    private ConquestMapFileWriter l_conqMapFileWriter;

    /**
     * Adapter constructor for setting conquest map file Writer.
     *
     * @param p_conqMapFileWriter conquest map file Writer
     */
    1 usage   Amanpreet
    public MapWriterAdapter(ConquestMapFileWriter p_conqMapFileWriter) {
        this.l_conqMapFileWriter = p_conqMapFileWriter;
    }


    /**
     * Adapter for writing to different type of map file through adaptee.
     *
     * @param p_gameState current state of the game
     * @param p_writer file writer
```

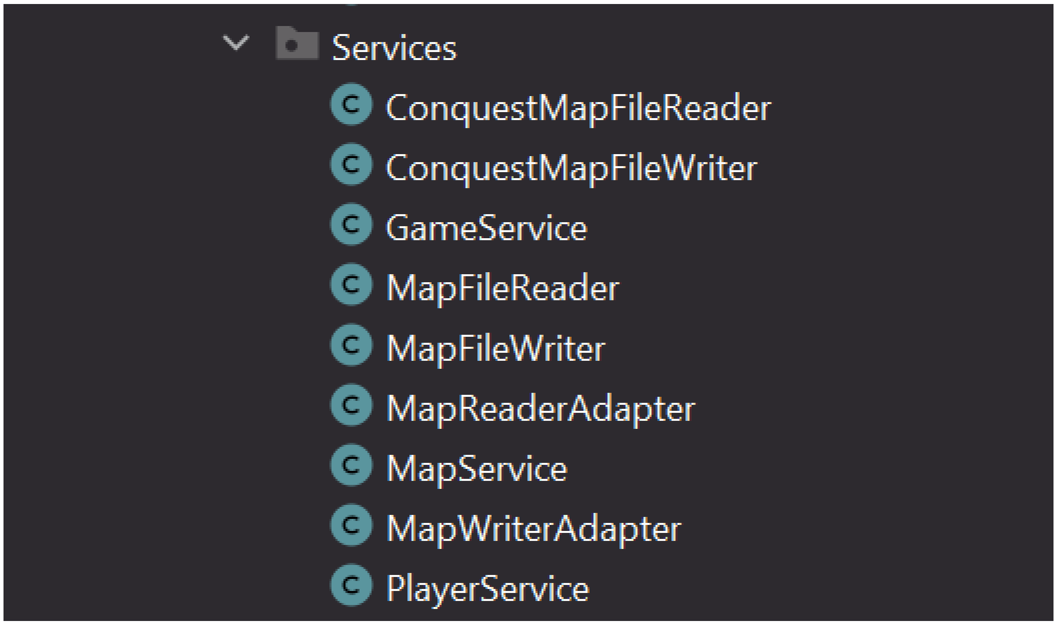
4) New Services Added for modularized this package

Before Refactor



```
Services
  MapService
  PlayerService
```

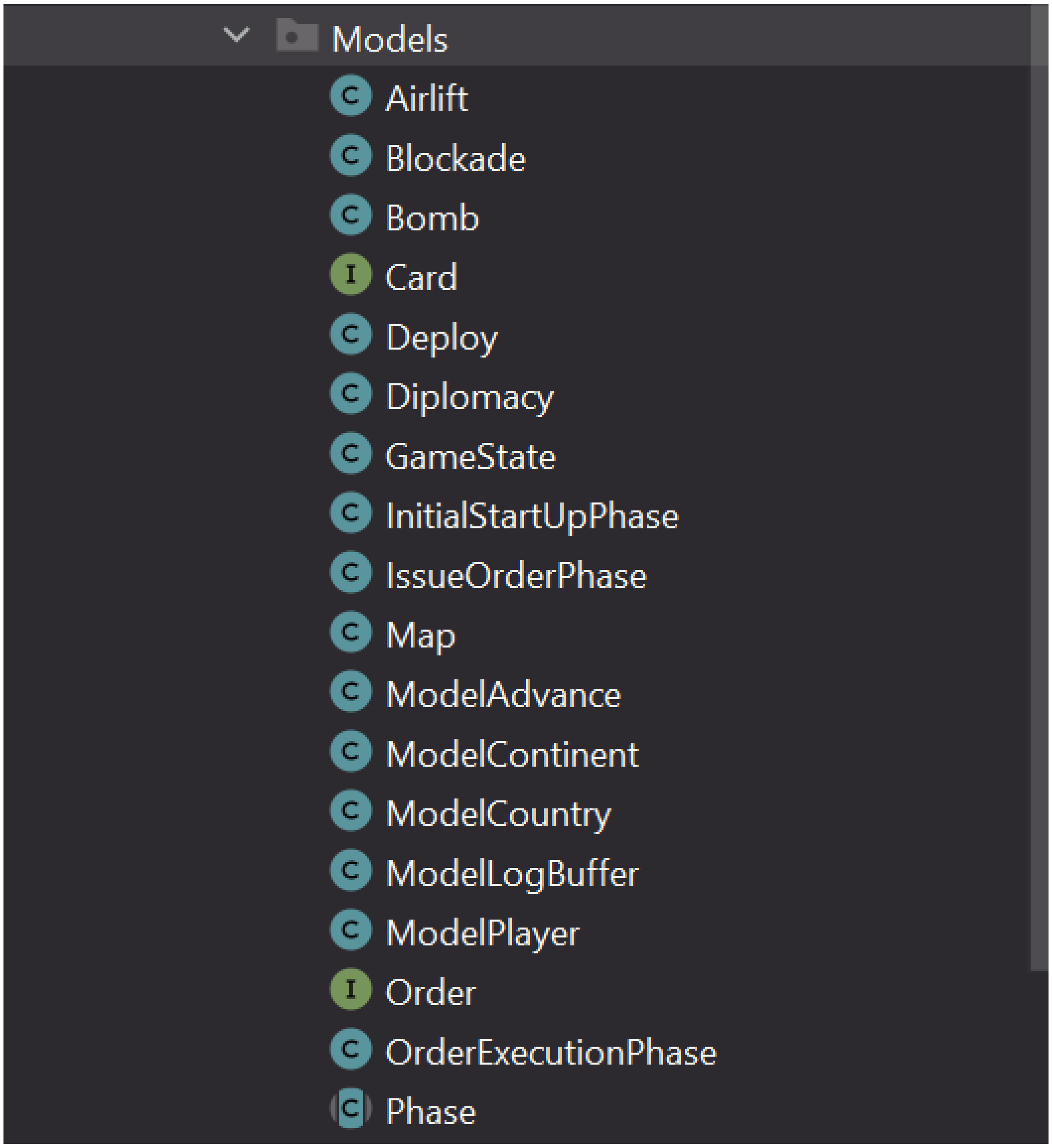
After Refactor



Services
ConquestMapFileReader
ConquestMapFileWriter
GameService
MapFileReader
MapFileWriter
MapReaderAdapter
MapService
MapWriterAdapter
PlayerService

5) New classes added catering to the requirement of new Models and better naming done

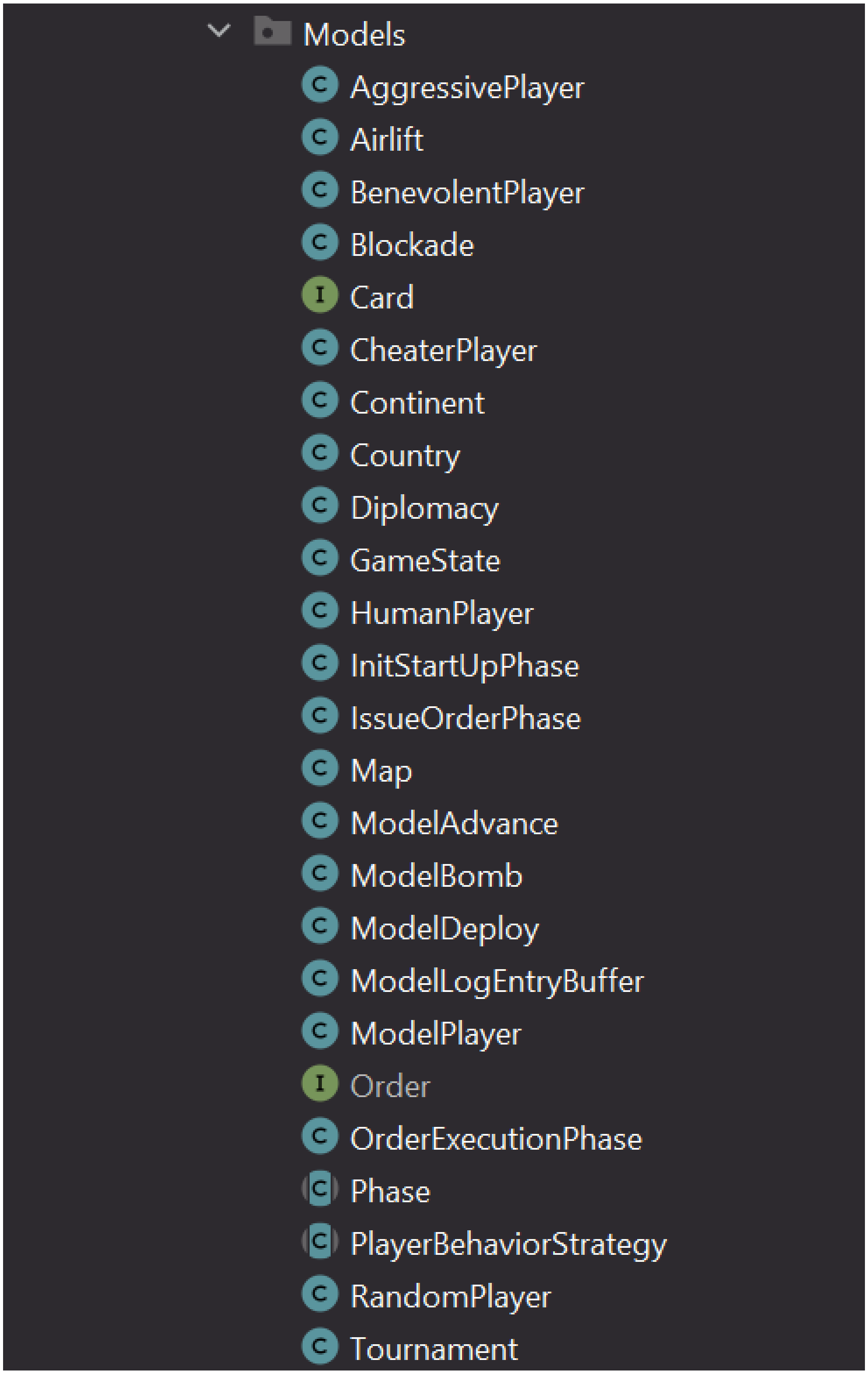
Before Refactor:



Models

- C Airlift
- C Blockade
- C Bomb
- I Card
- C Deploy
- C Diplomacy
- C GameState
- C InitialStartUpPhase
- C IssueOrderPhase
- C Map
- C ModelAdvance
- C ModelContinent
- C ModelCountry
- C ModelLogBuffer
- C ModelPlayer
- I Order
- C OrderExecutionPhase
- C Phase

After Refactor:



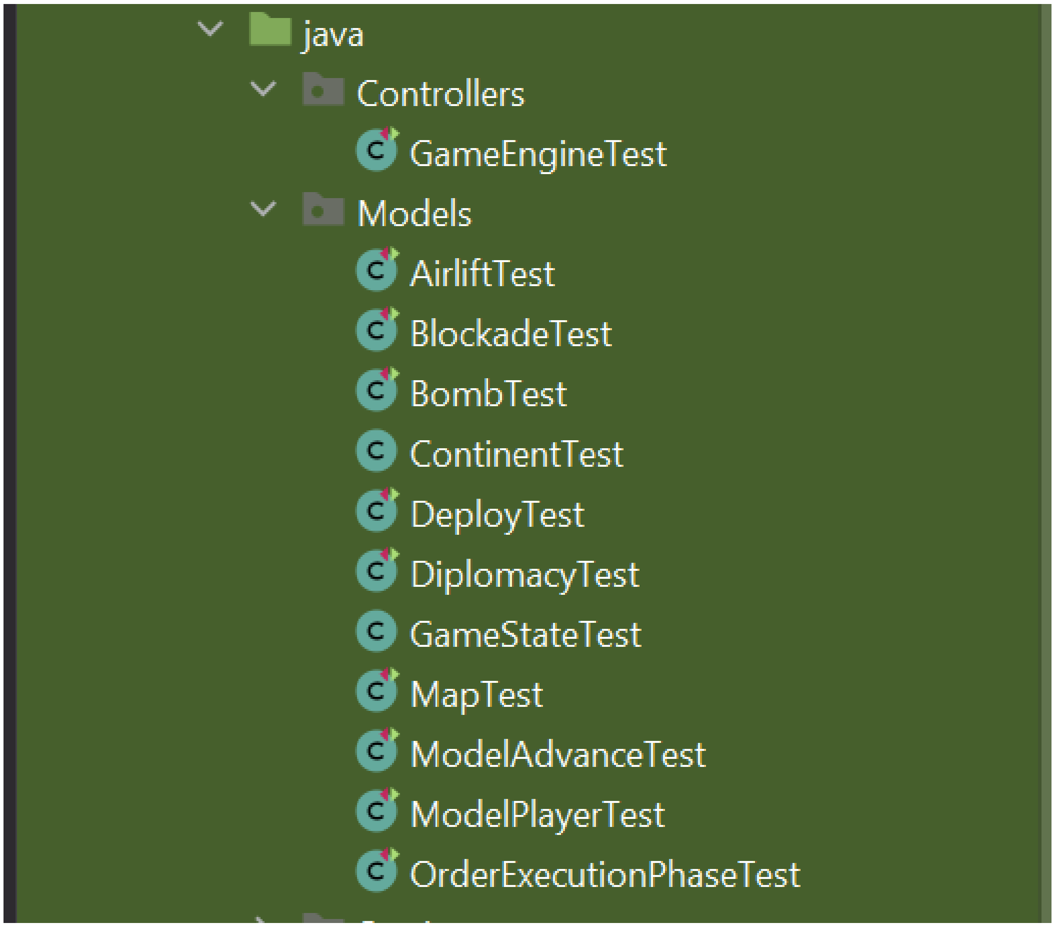
Models

C AggressivePlayer
C Airlift
C BenevolentPlayer
C Blockade
I Card
C CheaterPlayer
C Continent
C Country
C Diplomacy
C GameState
C HumanPlayer
C InitStartUpPhase
C IssueOrderPhase
C Map
C ModelAdvance
C ModelBomb
C ModelDeploy
C ModelLogEntryBuffer
C ModelPlayer
I Order
C OrderExecutionPhase
C Phase
C PlayerBehaviorStrategy
C RandomPlayer
C Tournament

6) More test cases added

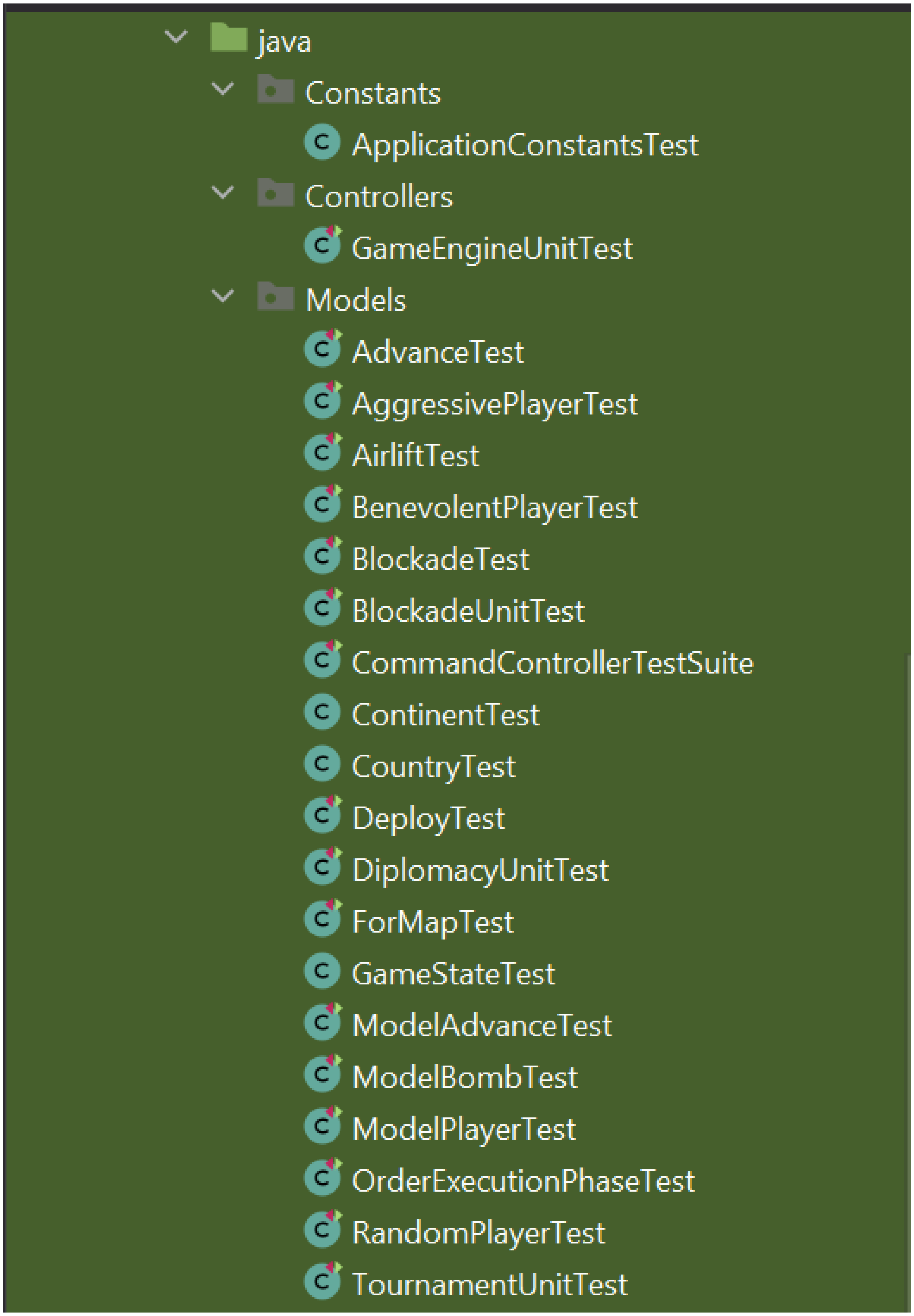
Before refactor

Test-cases were less so more test cases were added and test cases were mapped to the name of the actual model class.



```
java
  Controllers
    GameEngineTest
  Models
    AirliftTest
    BlockadeTest
    BombTest
    ContinentTest
    DeployTest
    DiplomacyTest
    GameStateTest
    MapTest
    ModelAdvanceTest
    ModelPlayerTest
    OrderExecutionPhaseTest
```

After refactor



- java
  - Constants
    - ApplicationConstantsTest
  - Controllers
    - GameEngineUnitTest
  - Models
    - AdvanceTest
    - AggressivePlayerTest
    - AirliftTest
    - BenevolentPlayerTest
    - BlockadeTest
    - BlockadeUnitTest
    - CommandControllerTestSuite
    - ContinentTest
    - CountryTest
    - DeployTest
    - DiplomacyUnitTest
    - ForMapTest
    - GameStateTest
    - ModelAdvanceTest
    - ModelBombTest
    - ModelPlayerTest
    - OrderExecutionPhaseTest
    - RandomPlayerTest
    - TournamentUnitTest