**Refactoring after Build 1:**

**Potential refactoring targets were identified on basis of some of the following factors**.

* To ease the further adding of new functionality.
* To clean up the code to reduce further possibility of introducing of bugs.
* Making the code readable and easy to understand.
* Simplifying the overall code structure and project architecture.

**List of potential refactoring targets.**

1. Methods from old *PlayerActions* class like - (Add/Remove players, assign Countries to players, assign armies to countries of players) is added into *PlaySetup* Class in State Pattern and *PlayerAction* class has been deleted.
2. Processes of all the *MapEditor* Commands has been moved from *GameController* class into *PostLoad* class.e.g. processing of editcountry commands.
3. Addition of **Observer pattern** to handle all the project output in one place.
4. Addition of **State Pattern** and moving the main-play and flow of execution phase wise, dedicated to separate classes.
5. Addition of Command **pattern** where each order has a separate class that implements in Order interface.
6. Change in package structure as we have removed the MVC pattern into a simpler structure where observer is only being used for new functionality of logging.
7. Moving “startGame” method from *Main.java* class to *GameEngine.java* class
8. Introducing abstract *Territory* class which is then extended by Country and Continent class for some same data members.
9. Introduction of “getTerritoryByName” method in map.java to get the territory that could be either country or continent using a single method. Previously, we were calling both country and continent arraylist separately to find territory.
10. *GameController* : Moving all the commands input to *GameEngine* class
11. Renaming method names in *ValidateMap*, *GameEngine* and other classes to a better name that can easily identify its purpose.
12. Moving the command input of order inside mainplay into *OrderCreation* class only, as these inputs are supposed to come in *OrderCreationPhase* only.
13. Creating a new Order Creation class for handling orders instead of implementing in *GameEngine* class
14. Creating a separate *OrderExecution* class dedicated for executing all the order one by one and reject those that are invalid.
15. Addition of getters and setter for Map in *GameEngine* class so that any class that has game-engine instance can also access the map and it’s current details.
16. Unit test of editcommands. For example: “editcountry” command input test method is moved from GameControllerTest.java to PostLoadTest.java. PostLoad.java is the state class that comes in operation in post load phase.

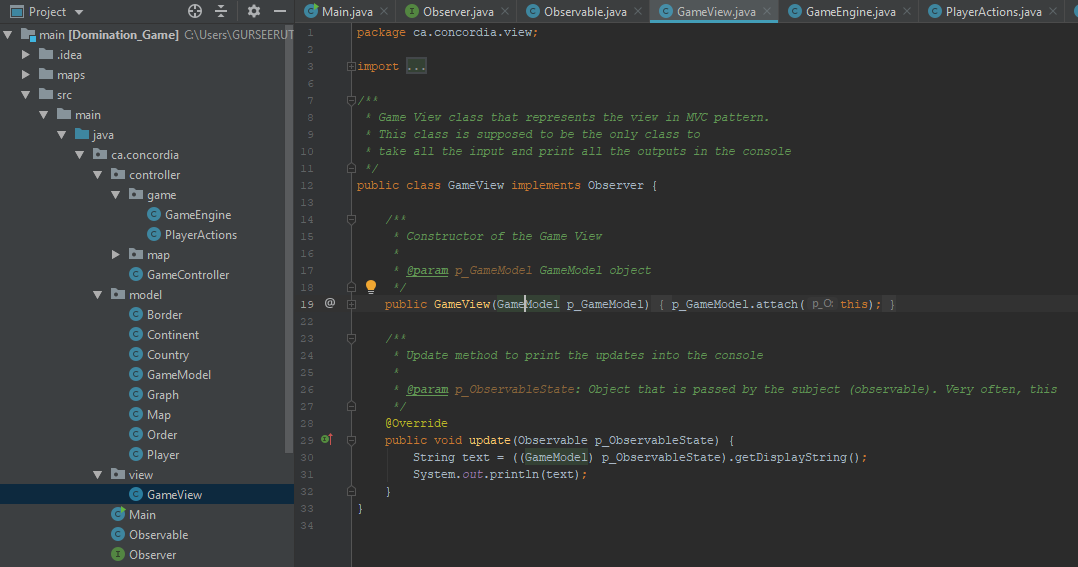
**List of refactoring to be made:**

1. Addition of Observer pattern to handle all the project output in one place.
2. Addition of State Pattern and moving the mainplay and flow of execution phase wise, dedicated to separate classes.
3. Addition of Command pattern where each order has a separate class that implements in Order interface.
4. Moving startGame method from Main.java to GameEngine.java class
5. Introducing an abstract class name Territory which is then extended by Country and Continent class for some common data members and methods.

**Refactoring Implemented along with all the Test classes.**

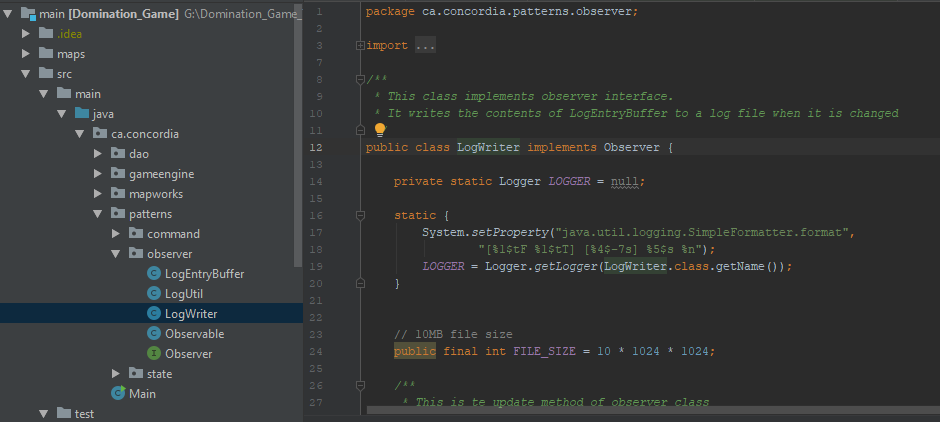
1. **In previous build**

GameView class Implements observer interface. It represents view in MVC pattern.

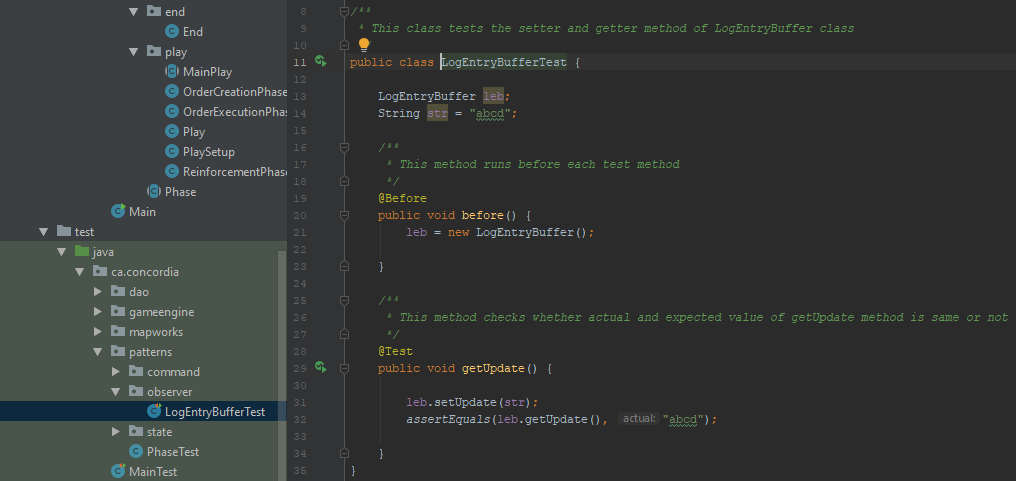


**Code after Refactoring:**

Changed package structure where MVC pattern is removed and observer is only being used in the new functionality that is logging.

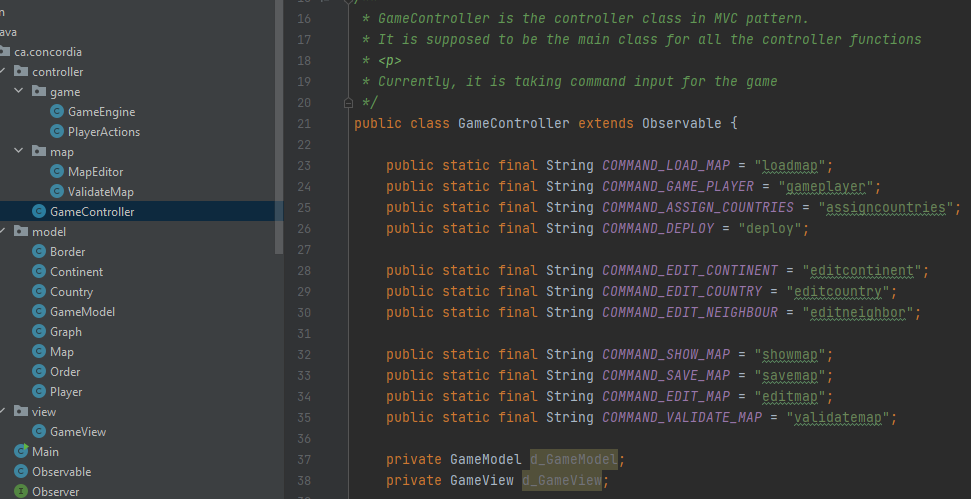


**Test class for Observer:-**

****

1. **In previous Build:**

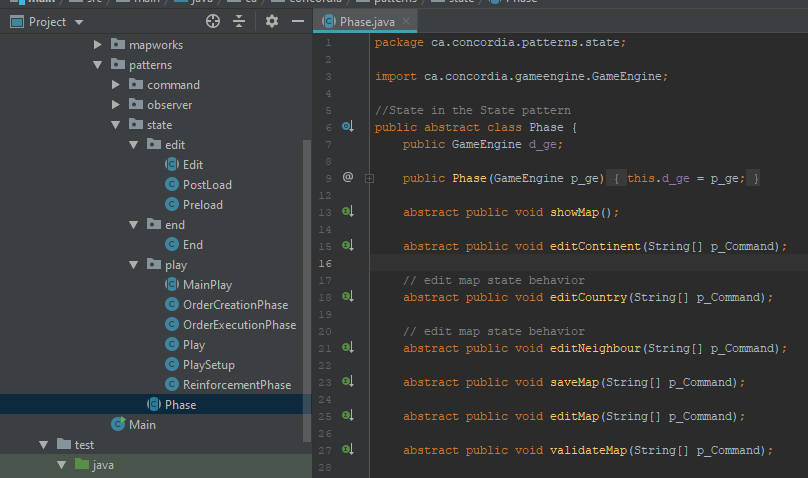
All the commands were implemented in *GameController* class and orders were created and executed in *GameEngine*



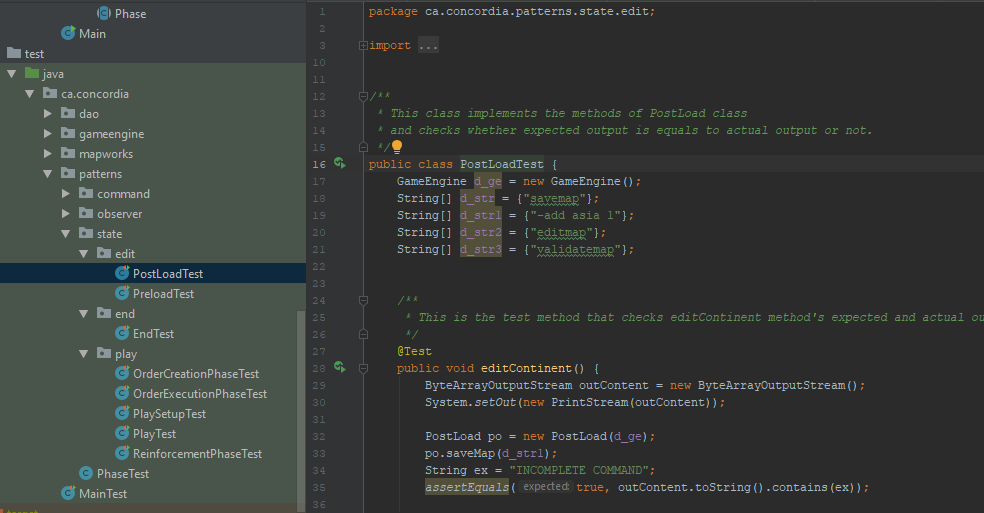
**Code after Refactoring:**

State pattern: edit, play and end commands with handling all the order creations and preload and post-load executions.

Introducing abstract Phase class in State Pattern to make it more object oriented implementation.

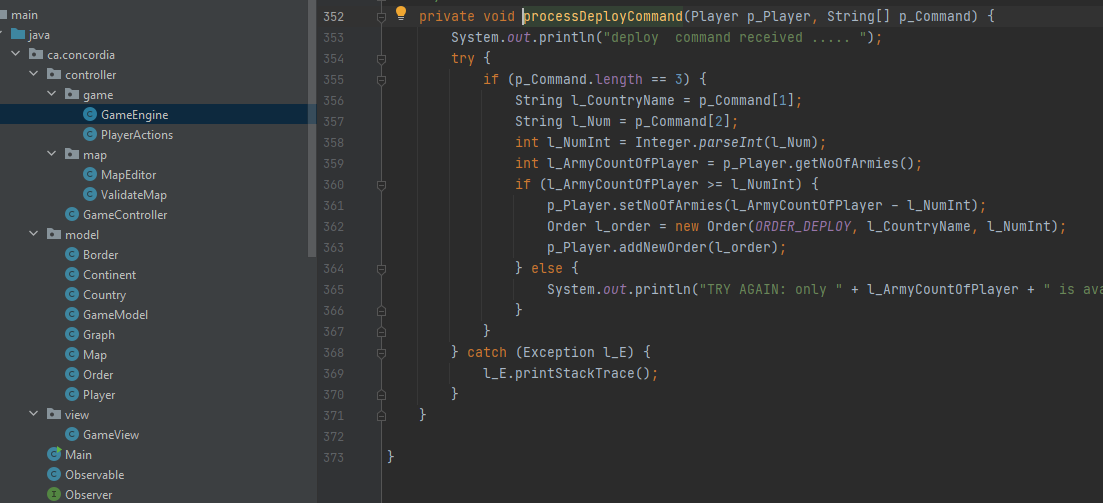


**Test classes for State Pattern**

****

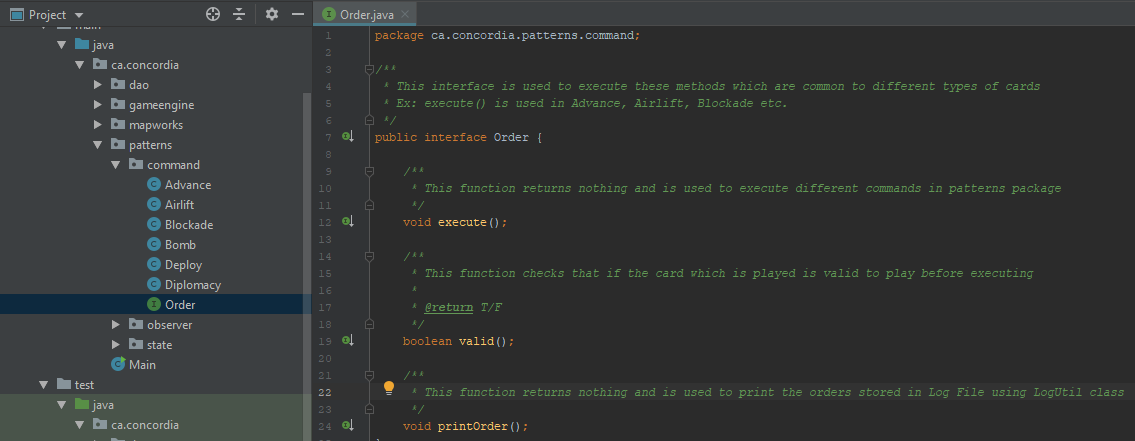
1. **In previous Build:**

The command such as deploy was being executed in the GameEngine class but is now being refactored to command pattern handling deploy order command along with more orders.

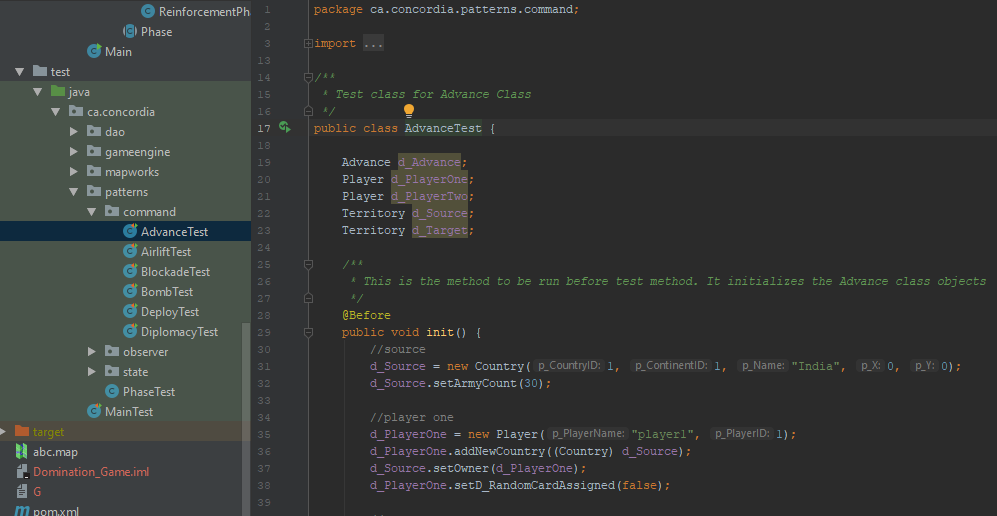


**Code after Refactoring:**

Command Pattern: Introducing a new interface named *Order,* which is then inherited by Advance, Airlift, Blockade, Bomb, Deploy, Diplomacy class and execution of these commands. So, implementation of existing “deploy” commands is moved to its dedicated command class Deploy now.

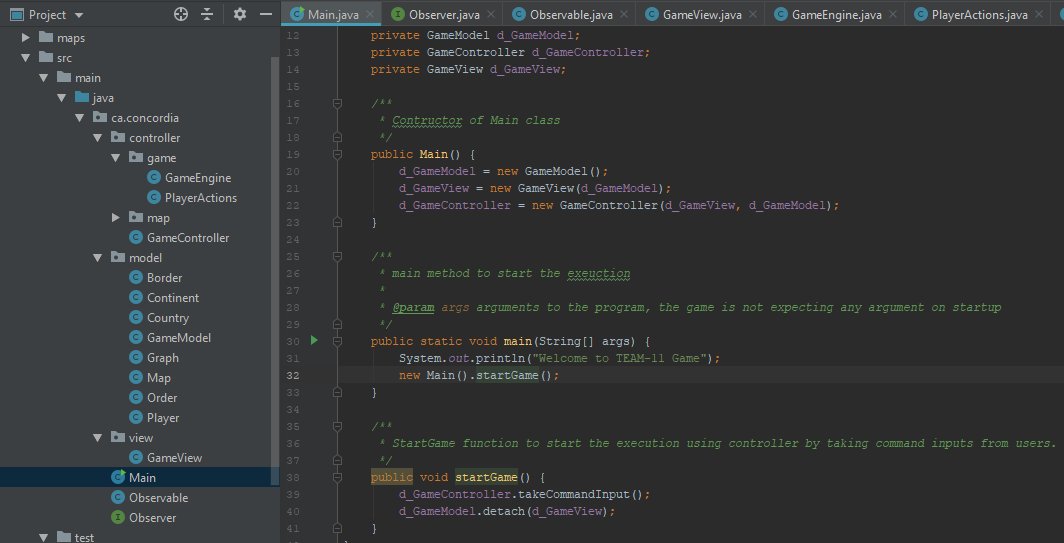


**Test classes for command pattern**

****

4. **Previous code:**

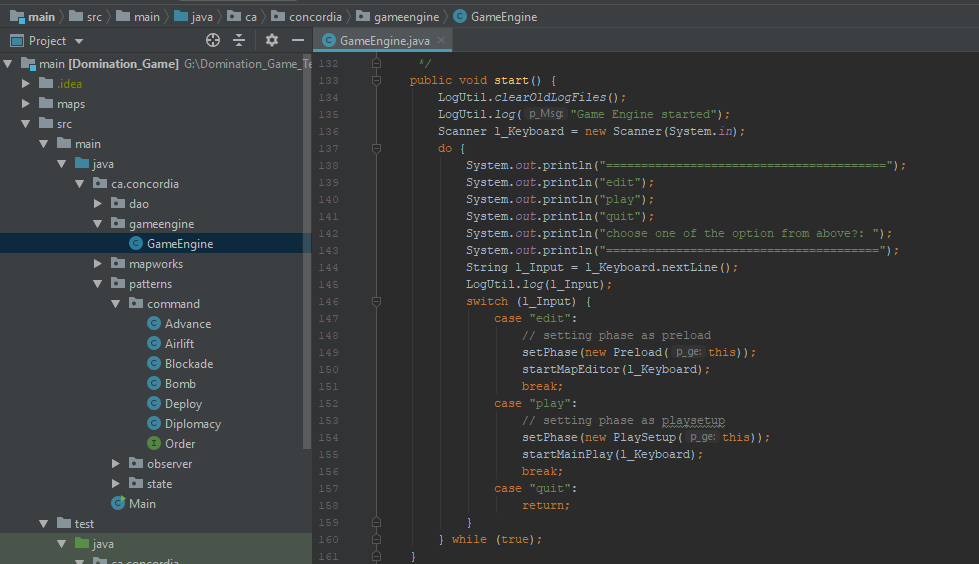
Main.java - startGame function to start the execution using controller by taking command inputs from user.



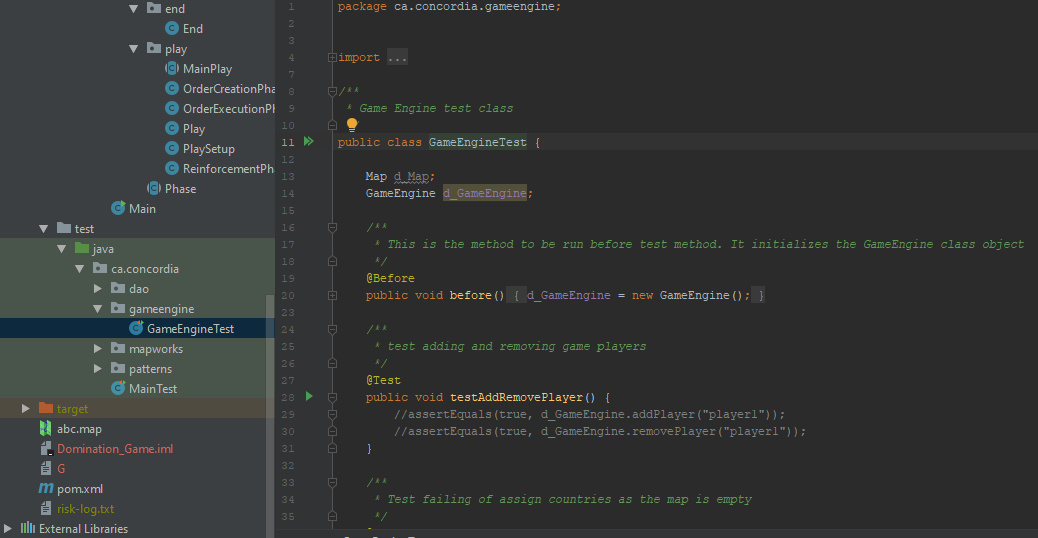
**Code after Refactoring:**

GameEngine.java - It takes input from a user to enter the phase.

Three phases are: Edit, Play, Quit

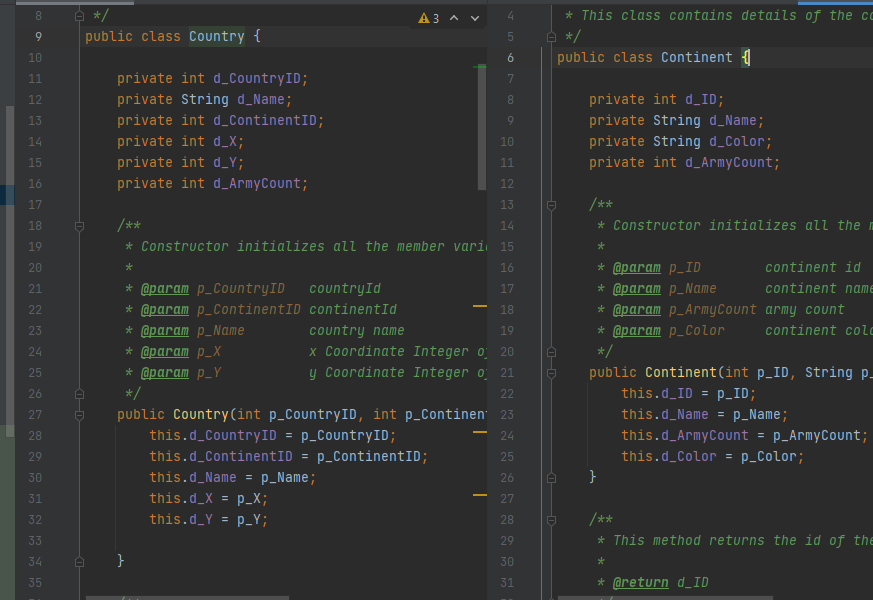


**Test class for GameEngine.java:**

****

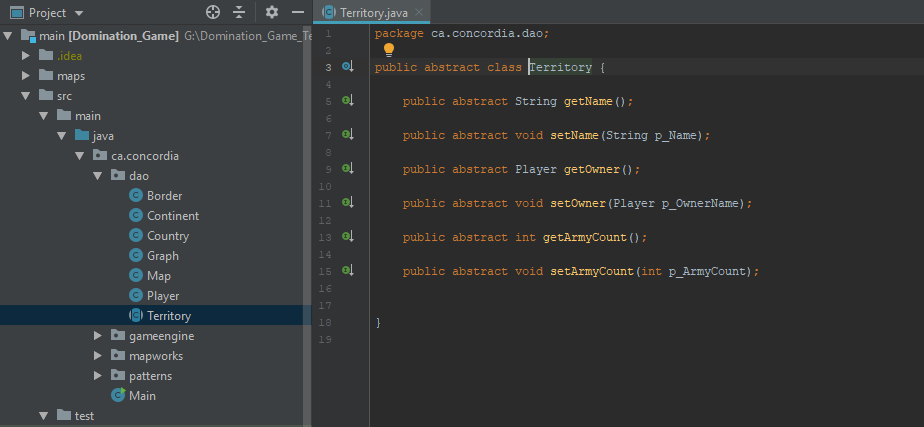
5. **Previous code:**

Initially, Country and Continent class had similar data members defined in both the class. Refactoring is done to introduce the abstract class that includes a common method in the parent allowing us to access the feature of both Country and Continent using the interface.

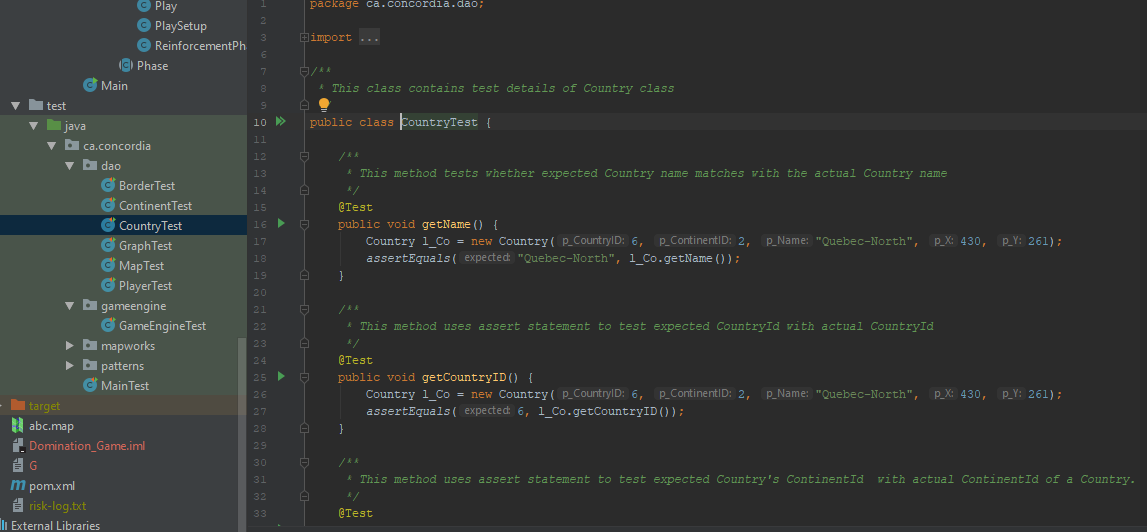


**Code after Refactoring:**

An abstract Territory class has been introduced which is then extended by Country and Continent class for some same data members.

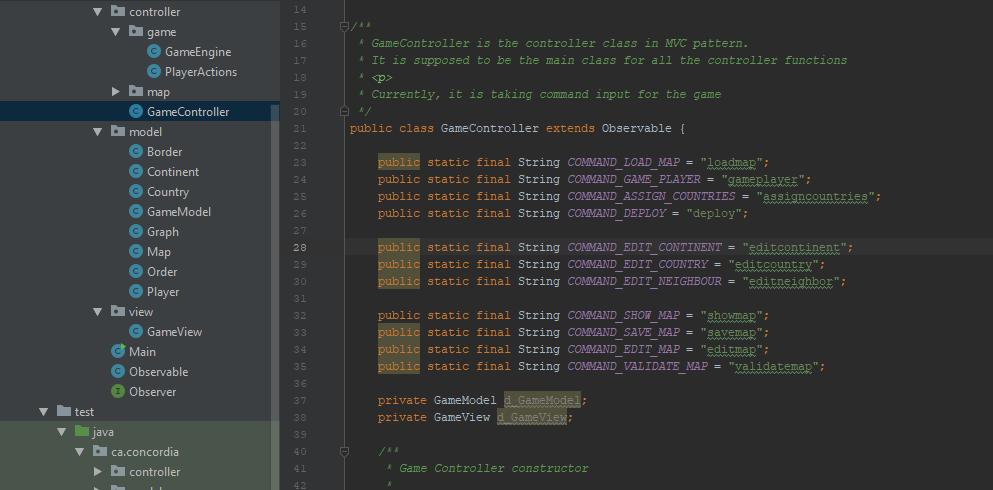


**Test classes:-**

****

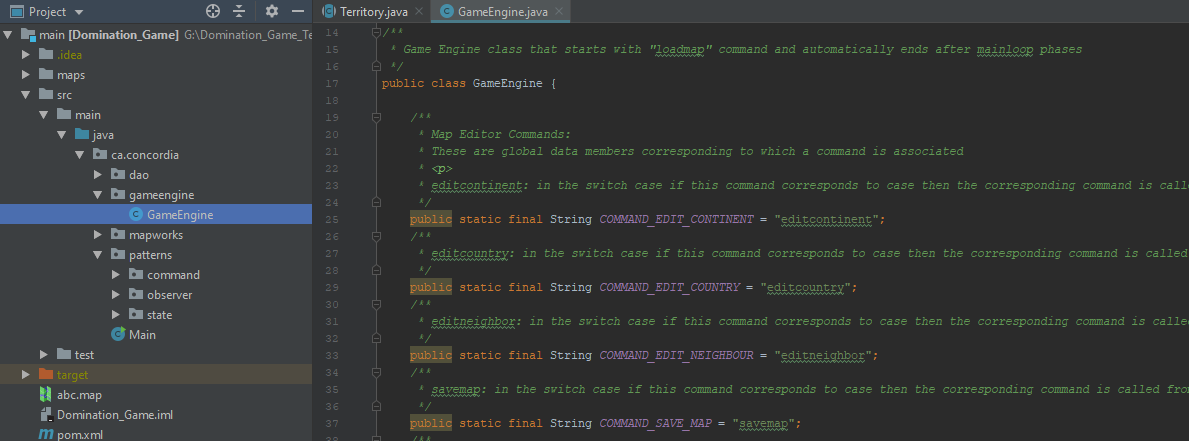
6. **Previous code:**

GameController.java - controller class in MVC pattern. It was supposed to be the main class for all the controller functions.



**Code after Refactoring:**

Moving all the commands input to GameEngine.java class. This starts with “loadmap” command and automatically ends after main-loop phases.



**Test class:**

