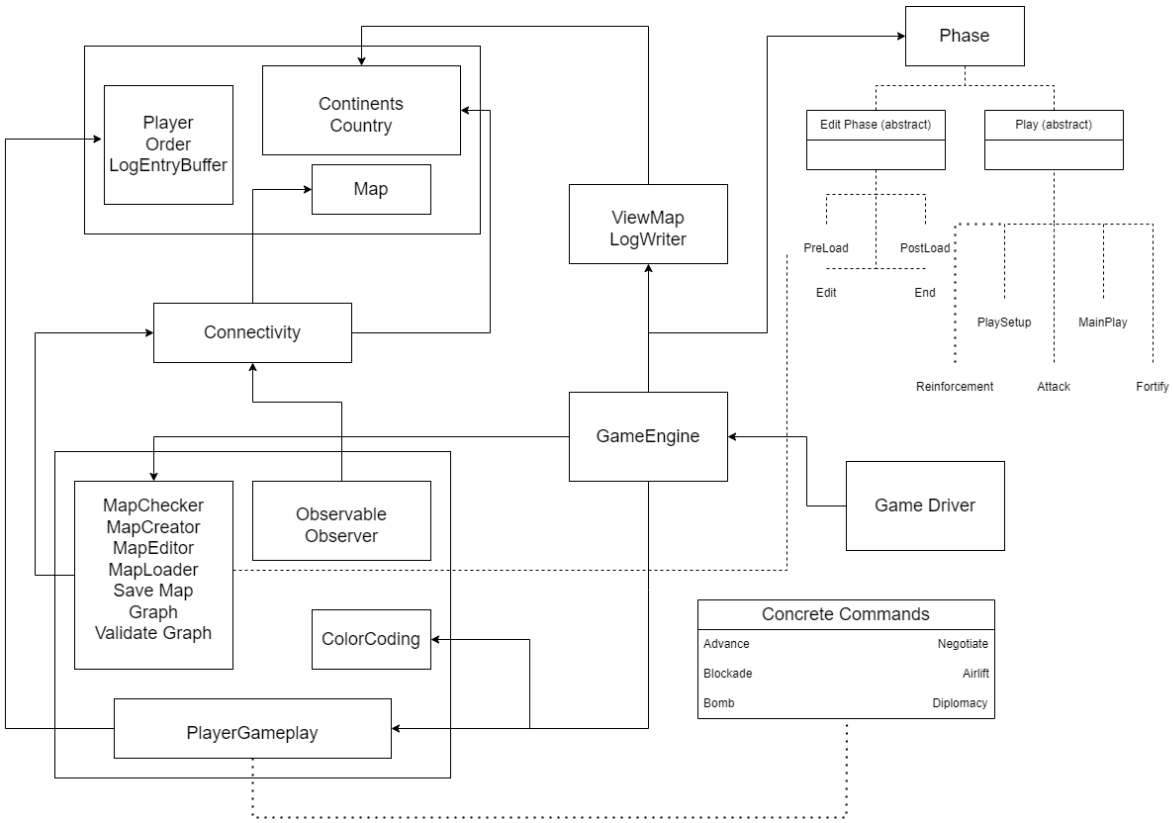

The following are the five primary modifications we made to our current code in build 2: -

1. State pattern implementation
2. Putting the command pattern into practise
3. Putting observer pattern into practise
4. Refactoring existing code.
5. Putting in place various player orders
    a. Advance
    b. Airlift
    c. Bomb
    d. Blocking
    e. Diplomacy

- State Pattern
    - ➢ GameEngine class is the context class in this case.
    - ➢ Phase class is the state class.
    - ➢ We primarily separate our existing EditPhase and Play classes into distinct phases in order to incorporate state patterns in our existing code.
    - ➢ There are different phases for Play class:
        - PlaySetup
        - MainPlay
        - Attack
        - Fortify

        - Reinforcement
    - ➢ There are different phases for EditPhase class:
        - PreLoad
        - PostLoad
        - Edit
        - End

- Command Pattern
    - ➢ Here the client is GameEngine class.
    - ➢ Invoker is an Order class.
    - ➢ Command is a PlayerGameplay class.
    - ➢ In order to incorporate command patterns into our current system, we primarily create six distinct concrete commands for varying ordering. the specific directive for the directive pattern as follows:
        - Advance
        - Airlift
        - Blockade
        - Bomb
        - Diplomacy
        - Deploy

- Observer Pattern
    - ➢ LogEntryBuffer and LogWriter are two classes that implement observer pattern.
    - ➢ Observer and Observable interfaces are used to provide definitions for the functions.

- Refactoring Targets for Build 2 -
    - ➢ State Design Pattern Implementation
    - ➢ Command Design Pattern Implementation
    - ➢ Create functions to obtain countryID from name and countryName from ID instead of repeatedly rewriting their functionalities
    - ➢ Change viewmap to include fields, Country Owner and armies currently deployed on country
    - ➢ Perform validating map upon loading instead of having user manually do it
    - ➢ Rename continent army value to continent bonus value
    - ➢ Maintain list of neutral countries (not owned by any player)
    - ➢ Connectivity object has to include data-member, ‘mapName’ as entered by user
    - ➢ Changed ‘savemap’ command functionality to ‘savemap filename’
    - ➢ Exceptions have to been moved to a separate package called ‘Exceptions’
    - ➢ Add Continent controlled by player within the model ‘player’
    - ➢ Reformat print messages of commands on the terminal
    - ➢ Maintain consistent color-coding of red for errors, green for successes on print statements across the entire program

- ➢ Renamed ‘validateMap’ in MapCheck.java to ‘checkIfMapNameExists’.
- ➢ Changed fileWriter for reading map file to BufferedReaader.

- The 5 chosen refactoring targets –
  1. State Design Pattern Implementation –
     a. The state pattern introduces phases such as Preload, Postload, PlaySetup, Reinforcement, Attack, Fortification, End. The context class of the State pattern is GameEngine class, and the State class is a new class named Phase.
     b. Tests implemented – phaseValidation, used for testing of successful transitions of different phases
  2. Command Design Pattern Implementation –
     a. The Command class in our case is the Order class, the Invoker Class in our case is the Player, and the Client class is our GameEngine.
     b. The orders are created as the player executes its issueOrder() method, and the nextOrder() method is for fetching orders from the player, and then the retrieved order is executed by calling the execute() method of the Order.
     c. Concrete Implementation of Abstract Order class - Advance, Airlift, Blockade, Bomb, Deploy, Diplomacy.
     d. Test implemented – Every attack phase command has its own set of validation tests
  3. Create functions to obtain countryID from name and countryName from ID instead of repeatedly rewriting their functionalities
     a. getCountryFromID function is now used to get countryName from a given countryID
     b. getNameFromID function is now used to get countryID from a given countryName
     c. Tests implemented – checkCountryFromID, checkNameFromID has been implemented
  4. Change viewmap to include fields, Country Owner and armies currently deployed on country
     a. Since each country can be controlled by individual players, it is important to display to the user/player what country he/she controls
     b. Since each country can have armies deployed on them, we display the number of armies deployed on each country in viewmap
     c. Tests implemented – Individual test cases written for assigning/removing player’s control from countries and modifying army count
  5. Perform validating map upon loading instead of having user manually do it
     a. User had to manually validate the map after loading a new map after launching the game
     b. From this change, the loaded map is automatically validated to show if the map is a valid map or not
     c. Tests implemented – verified in the test case, phaseValidation