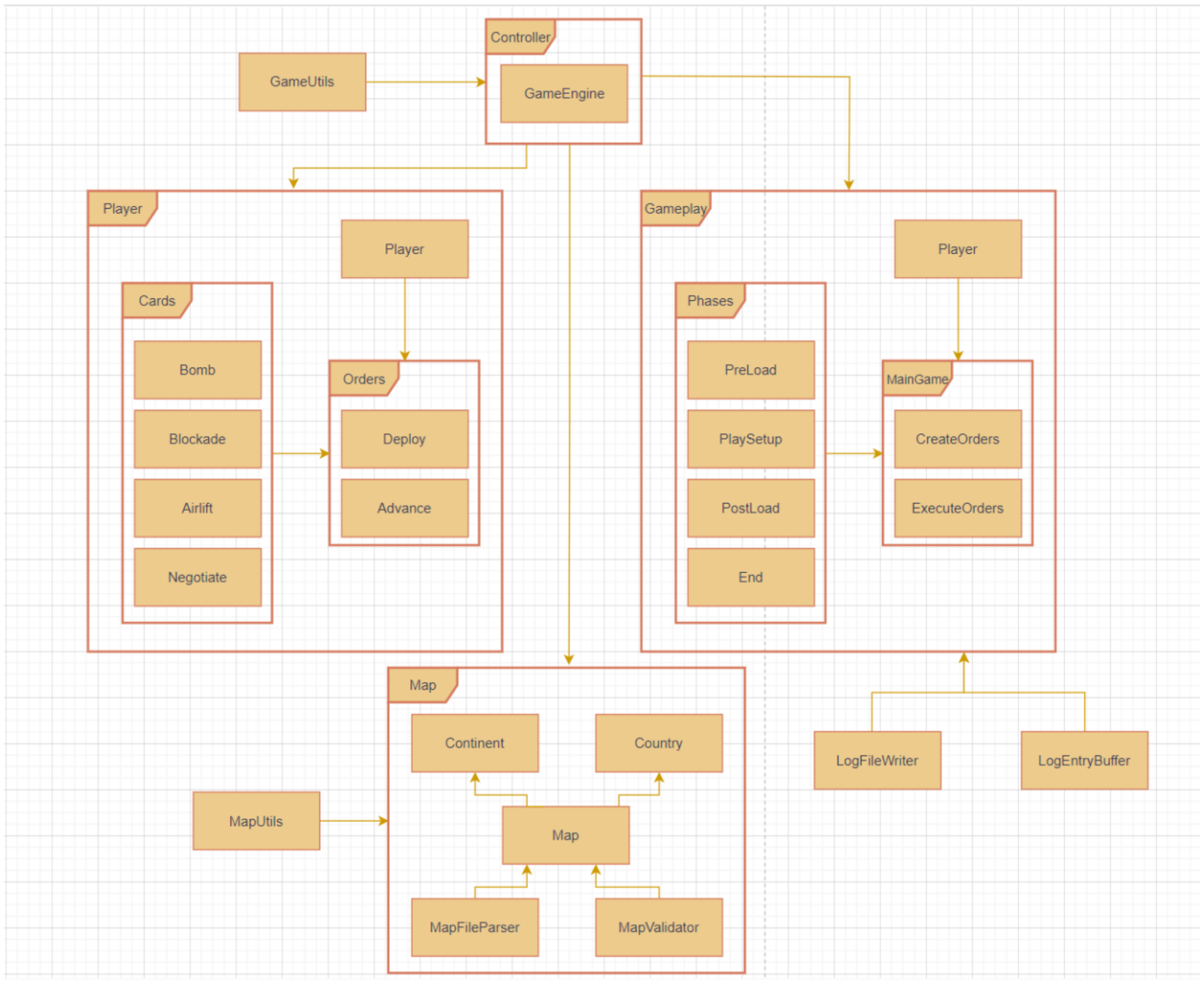
Team U6: Build 2

Architectural Diagram:



In the second build of the project, we have taken care of the following:

Implementation of Various orders for the players, including-

- Advance
- Airlift
- Bomb
- Blockade
- Diplomacy

Refactoring:

We have refactored our code to use the State Pattern, command pattern and observer pattern.

State Pattern:

A new class phase has been created which has been implemented as an abstract class, to divide the map editor functions and game play into phases. The Phase class contains a reference to the State of the GameEngine so that the state object can change the state of the GameEngine to transition between states. There are three main phases: startup, issue order, and order execution. Each of these represent a state in the game play:

- Edit phase (abstract)
    - Preload
    - Postload
- Play (abstract)
    - PlaySetup
    - MainPlay
        - CreateOrders
        - ExecuteOrders
- End

Command Pattern:

Command Pattern has been used to implement orders. The Command class is the Order class, the Invoker Class is the Player, and the Client class is the GameEngine. The orders are created as the player executes its issue_order() method, and the orders are executed when the GameEngine gets the Player's orders from the Players using the next_order() method, then executes the orders by calling the execute() method of the Order.

Observer Pattern:

We have implemented the game log file using the observer pattern. We have a LogEntryBuffer Class that extends Observable and records the corresponding logs for various stages in game. We also have a LogFileWriter Class that implements Observer Class and updates the Log File with logs from LogEntryBuffer Class.