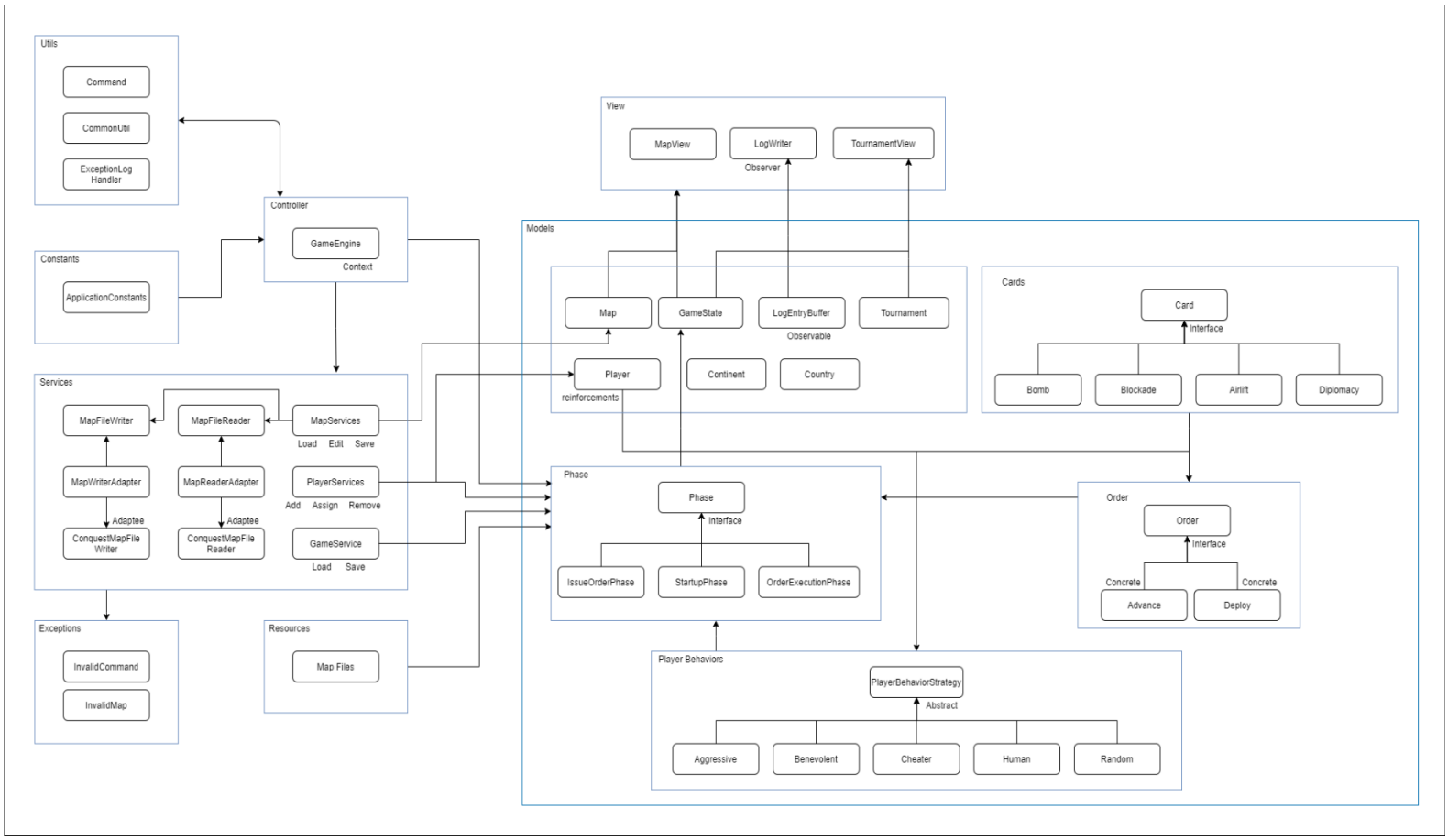
Build #3 – Architecture Diagram



A. Models

- Player: This class manages all the continents, countries, the orders given, and armies owned by each Player object.

- Order: This class handles the order action, the source and destination country and the number of armies to move.

- GameState: Manages the Player and Order objects and maintains the list of orders which are in the list (yet to be executed).

- Map: Manages the Country and Continent objects and Validates the Map by checking the connectivity between the existing countries and continents.

- Continent: Manages all the Continents along with the bonus value of each continent. Also, Country object is linked with the Continent object to map all the countries that exist in the Continent.

- Country: Manages all the Countries along with the number of armies present on each of them and the Countries adjacent to it.

- LogEntryBuffer: **(Observer Pattern Implementation)** This class records the corresponding logs for various stages in the game.

- Tournament: Tournament class handles the tournament gameplay which will be played automatically by different Player Strategies

- Phase: **(State Pattern Implementation)**
    i. IssueOrderPhase: Manages the issue order phase of the gameplay and handles the commands accordingly.
    ii. StartupPhase: Manages the startup phase of the gameplay and handles the commands accordingly.
    iii. OrderExecutionPhase: Manages the order execution phase of the gameplay and handles the commands accordingly.

- Card:
    i. Bomb: Manages the bomb card logic wherein the target country loses half of their army units when this card this used by the player on the enemy territory.
    ii. Blockade: Manages the blockade card logic wherein one of the player's territory becomes neutral and triples the armies on the same territory.
    iii. Airlift: Manages the airlift card logic wherein the player can move any number of armies from one of his territories to another, even if they are not adjacent.
    iv. Diplomacy: Manages the diplomacy card logic wherein the current player and the enemy player will not be able to attack each other until the end of the existing turn.

- Order: **(Command Pattern Implementation)**
    i. Advance: Advancing the armies of the player from one territory to another territory. Advance order is performed only if the target and source territories are adjacent to each other.
    ii. Deploy: Deploying some of the armies to one of the current player's territories.

- PlayerBehaviorStrategy: **(Strategy Pattern Implementation)**
    i. AggressivePlayer: This class implements and handles the main logic of Aggressive Player who will gather all his armies, attacks from his strongest country and deploys armies to maximize his forces on one country.
    ii. BenevolentPlayer: This player will solely focus on defending his territories and will never attack.

iii. CheaterPlayer: Cheater Player will initiate attacks during the issue order phase and will double his armies on the territories that consists of enemy neighbours.
iv. HumanPlayer: This class manages the logic of receiving commands from the user and playing the game accordingly.
v. RandomPlayer: This class implements the logic of Random Player who will randomly choose a country to attack from, using random number of armies and then attacking any random enemy country.

B. View

- MapView: Displays the current state of the Map when user selects *showmap* command in the GamePlay State and displays the overall existing Map while in EditMap phase.

- LogWriter: Writes logs to the log file which consists of all the logs derived from the LogEntryBuffer Model Class.

- TournamentView: Displays the whole Tournament Gameplay.

C. Controller

- GameEngineController: The main command handling logic is controlled from this class.

D. Services

- MapServices: The main logic to handle the state of the Map is handled here, along with map load, edit and save functionalities.

- PlayerServices: It enables the services related to Player – add, remove, assign continent, countries and armies.

- GameServices: Handles the services related to game such as save and load.

- MapWriterAdapter: **(Adapter Pattern Implementation)**
  i. MapFileWriter: This class is used to create the map files.
  ii. ConquestMapFileWriter: Manages the creating and writing operations on Conquest Map files.

- MapReaderAdapter: **(Adapter Pattern Implementation)**
  i. MapFileReader: This class is used to parse the map files.
  ii. ConquestMapFileReader: Manages the reading and parsing operations on Conquest Map files.

E. Utils

- Command: Takes and formats the commands given by the players.
- CommonUtil: Handles the input files.
- ExceptionLogHandler: Handles the exceptions that are related to Logs which are not caught by the try/catch block.

F. Constants

- ApplicationConstants: All the static members are defined here and can be used anywhere in the application at any point.

G. Exceptions

- InvalidCommand: Displays meaningful message to the player when invalid command is entered.
- InvalidMap: Displays meaningful message to the player when invalid map is created.

H. Resources

- Map Files: All the .map files, existing and newly created by users are stored and managed here.