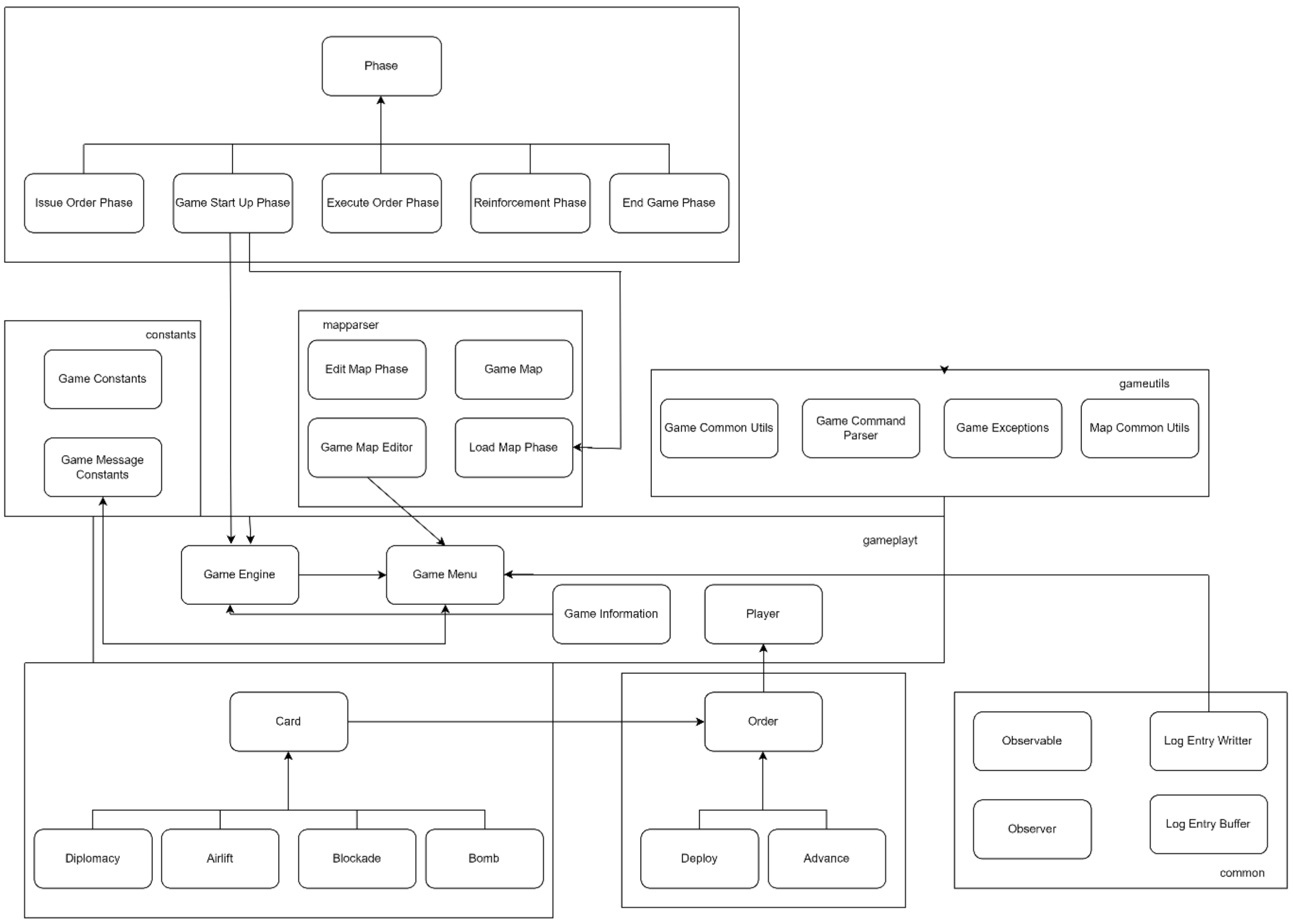
ARCHITECTURE DIAGRAM



Number of packages: 6

- common
- constants
- gamemenu
- gameplay
- gameutils
- mapparser

Command Pattern:

Order interface is responsible for executing the orders of the player. The Command pattern is applied with Order as the common interface, and DeployOrder and AdvanceOrder as concrete classes. The Invoker is responsible for queuing and executing these orders.

- DeployOrder: Class represents an order to deploy a certain number of armies to a specific country. This class extends Order.
- AdvanceOrder: Class deals with advancing of the armies for an attack on the desired country. This class extends Order.

The enum class Card distributes random cards to the players when the conditions are met – bomb, blockade, airlift, diplomacy. It is an enumeration.

- Bomb: The class deals with bombing of the countries using the bomb card, resulting in the army count halving. This class extends from Order.
- Blockade: The class is responsible for the blockade action, involves the army count to be multiplied by 3. This class extends from Order.
- Airlift: The class deals with airlifting of the armies to any other countries of the opponent chosen. This class extends from Order.
- Diplomacy: The class deals with the negotiation phase between the chosen players. This class extends from Order.

Observer Pattern:

The observer pattern is implemented using the LogEntryBuffer and LogEntryWritter classes as Observable and Observers respectively.

- Observable and Observer: Both being abstract classes with observable containing the notifyObservers(), addObservers() method and the observer containing the update() method.
- LogEntryBuffer – Extends Observable and implements the functionality of putting together the log messages acting as a buffer, before writing it into a file.
- LogEntryWritter – Extends Observer and implements the functionality of writing the logs into a file.

LogEntryBuffer acts as the observable, and LogEntryWriter acts as the observer. LogEntryWriter registers with LogEntryBuffer to receive updates, and when a new log entry is added to the buffer, it is automatically notified and processes the log entry.

State Pattern:

It is an interface having templates for executing different phases of the warzone game.

Phase: An Abstract class that has nextPhase() and executePhase() abstract methods that are extended by all Phase related classes listed below.

Map Edit Phase:

- LoadMapPhase: The class handles loading of the map, in terms of the phase. The Gamemap class is executed once the LoadMapPhase is executed.
- EditMapPhase: The class handles editing of the map, in terms of the phase. The GamemapEditor class is executed once the EditMapPhase is executed.

Gameplay Phase:

- ReinforcementPhase: This class is responsible for assigning reinforcement armies to players based on the number of countries they have conquered. It extends Phase.
- IssueOrderPhase: This class creates a deploy order object on the player's list of orders, then reduce the number of armies in the player's pool of armies. It extends Phase.
- ExecuteOrderPhase: The class deals with the execution of the orders issued by the players during the game.
- GameStartupPhase: Class Game StartUp Phase contains current phase, next phase, and current game information for an instance. It extends Phase.
- EndGamePhase: This is the final phase of the game before the game terminates. It extends Phase.

gameutils:

- GameCommandParser: This class is responsible for parsing user input commands into primary commands and their details.
- GameCommonUtils: This class provides utility methods for operations.
- GameException: The class handles exceptions.
- MapCommonUtils: This class reads a map file and extracts a specific section of the file based on starting and ending keywords.

mapparser:

- GameMap: The class loads the game map file that is chosen for play.
- GameMapEditor: The class is used to initialize map editing phase - which consists of
  editing, or creating a new map.

gameplay:

- GameInformation: This class is used to set and retrieve the current phase, current map and other information of the game at that instant.
- GameEngine: This class is responsible for executing the entire game process.

gamemenu:

- GameMenu: The GameMenu class is the entry point of the game. This is where the main() resides. From here the player has the option to choose between map editing or game play options.