COURSE: SOEN 6441:ADVANCED PROGRAMMING PRACTICES

SUPERVISOR: JOEY PAQUET

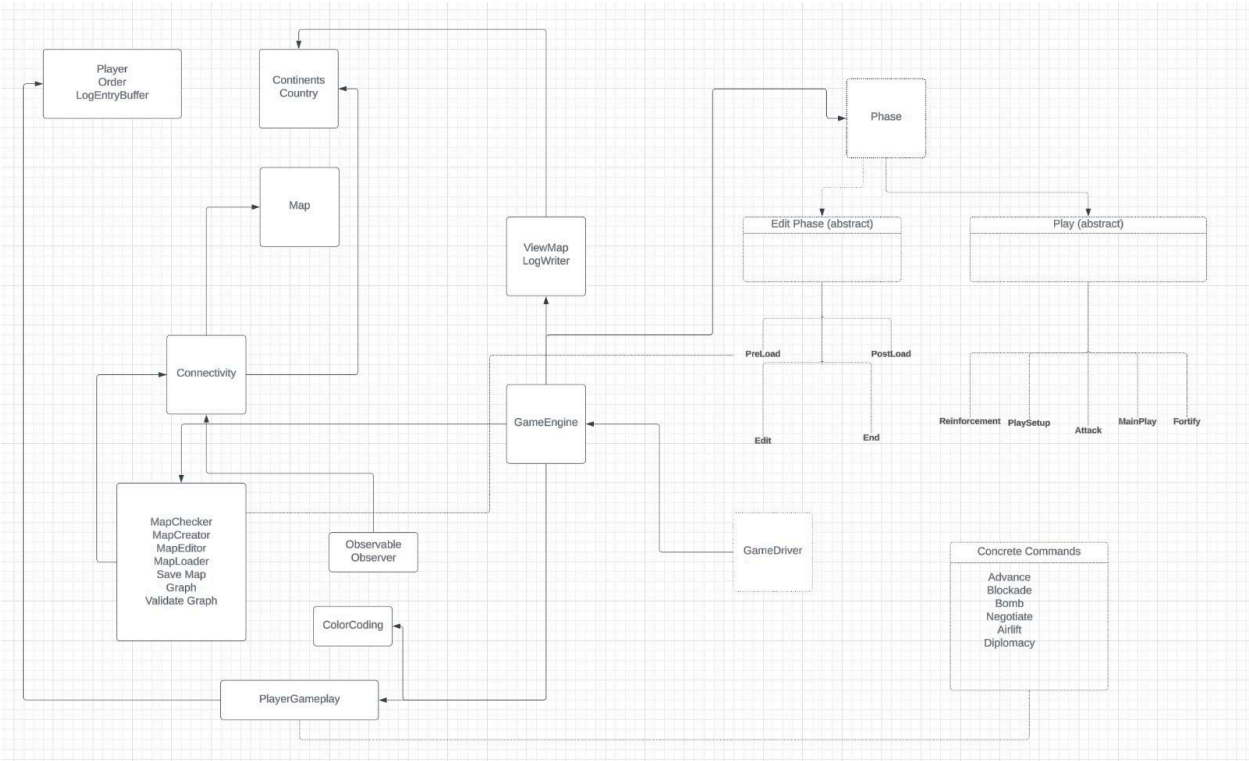
PROJECT: WARZONE GAME BUILD 1

TEAM 16: ARCHITECTURAL REPRESENTATION

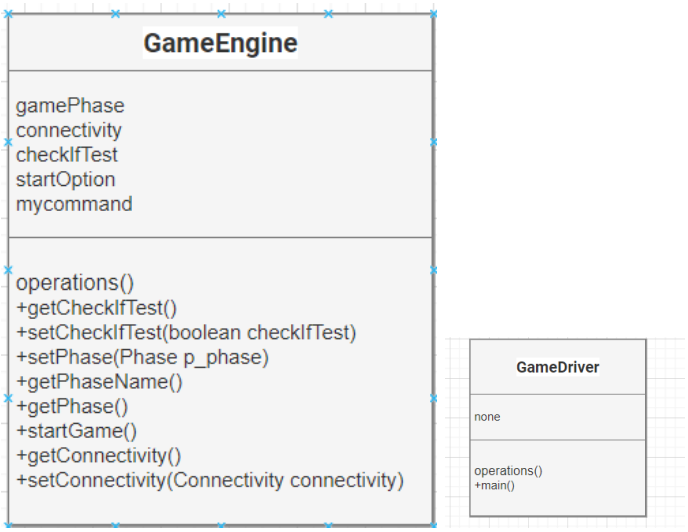
TEAM MEMBERS:

1. Sanjay Bhargav Pabbu
2. Piyush Gupta
3. Blesslin Jeba Shiny Augustin Moses
4. Mahfuzzur Rahman
5. Susmitha Mamula
6. Poojitha Bhupalli

ARCHITECTURAL DIAGRAM



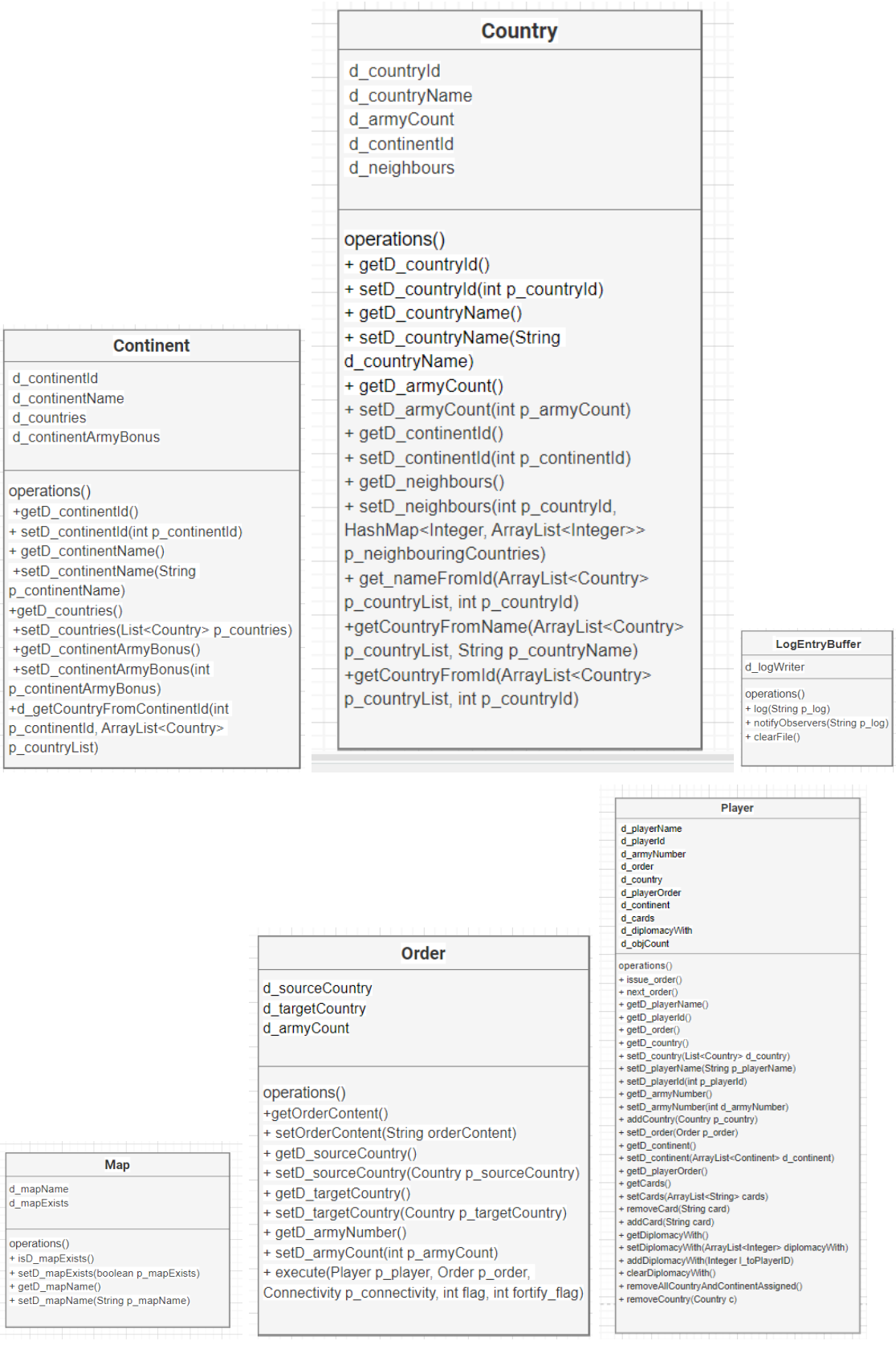
1. Controllers



a. GameDriver: This class is responsible for initiating the game engine.

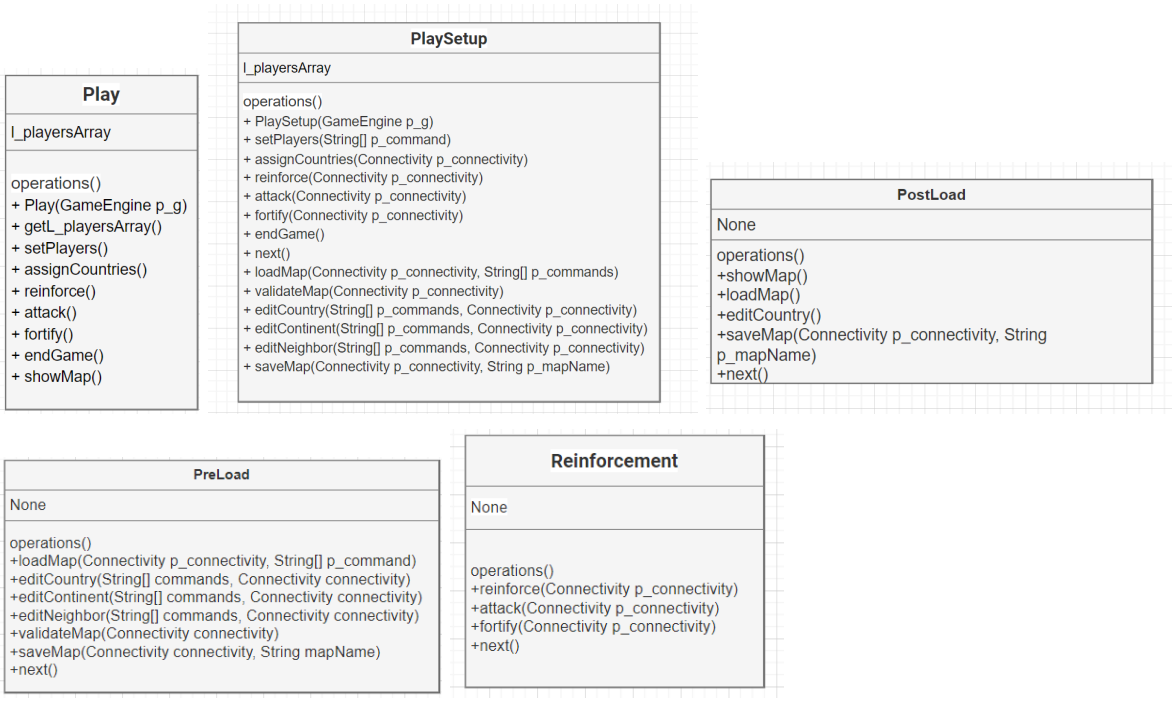
b. GameEngine: This class is the main module and it manages the whole game engine and its phases.

2. Models


Country
d_countryId
d_countryName
d_armyCount
d_continentId
d_neighbours
operations()
+ getD_countryId()
+ setD_countryId(int p_countryId)
+ getD_countryName()
+ setD_countryName(String d_countryName)
+ getD_armyCount()
+ setD_armyCount(int p_armyCount)
+ getD_continentId()
+ setD_continentId(int p_continentId)
+ getD_neighbours()
+ setD_neighbours(int p_countryId, HashMap<Integer, ArrayList<Integer>> p_neighbouringCountries)
+ get_nameFromId(ArrayList<Country> p_countryList, int p_countryId)
+getCountryFromName(ArrayList<Country> p_countryList, String p_countryName)
+getCountryFromId(ArrayList<Country> p_countryList, int p_countryId)
Continent
d_continentId
d_continentName
d_countries
d_continentArmyBonus
operations()
+getD_continentId()
+ setD_continentId(int p_continentId)
+ getD_continentName()
+setD_continentName(String p_continentName)
+getD_countries()
+setD_countries(List<Country> p_countries)
+getD_continentArmyBonus()
+setD_continentArmyBonus(int p_continentArmyBonus)
+d_getCountryFromContinentId(int p_continentId, ArrayList<Country> p_countryList)
LogEntryBuffer
d_logWriter
operations()
+ log(String p_log)
+ notifyObservers(String p_log)
+ clearFile()
Player
d_playerName
d_playerId
d_armyNumber
d_order
d_country
d_playerOrder
d_continent
d_cards
d_diplomacyWith
d_objCount
operations()
+ issue_order()
+ next_order()
+ getD_playerName()
+ getD_playerId()
+ getD_order()
+ getD_country()
+ setD_country(List<Country> d_country)
+ setD_playerName(String p_playerName)
+ setD_playerId(int p_playerId)
+ getD_armyNumber()
+ setD_armyNumber(int d_armyNumber)
+ addCountry(Country p_country)
+ setD_order(Order p_order)
+ getD_continent()
+ setD_continent(ArrayList<Continent> d_continent)
+ getD_playerOrder()
+ getCards()
+ setCards(ArrayList<String> cards)
+ removeCard(String card)
+ addCard(String card)
+ getDiplomacyWith()
+ setDiplomacyWith(ArrayList<Integer> diplomacyWith)
+ addDiplomacyWith(Integer I_toPlayerID)
+ clearDiplomacyWith()
+ removeAllCountryAndContinentAssigned()
+ removeCountry(Country c)
Order
d_sourceCountry
d_targetCountry
d_armyCount
operations()
+getOrderContent()
+ setOrderContent(String orderContent)
+ getD_sourceCountry()
+ setD_sourceCountry(Country p_sourceCountry)
+ getD_targetCountry()
+ setD_targetCountry(Country p_targetCountry)
+ getD_armyNumber()
+ setD_armyCount(int p_armyCount)
+ execute(Player p_player, Order p_order, Connectivity p_connectivity, int flag, int fortify_flag)
Map
d_mapName
d_mapExists
operations()
+ isD_mapExists()
+ setD_mapExists(boolean p_mapExists)
+ getD_mapName()
+ setD_mapName(String p_mapName)

a. **Continent**: The continent class is responsible for providing the geographical continents to the gameplay map. It defines a continent with its ID, name, list of countries, and army bonus.
b. **Country**: The Country class represents the country in which the game is supposed to be played. Additionally, it defines a country with its ID, name, army count, continent ID, and neighboring countries.
c. **LogEntryBuffer:** It handles logging of game events to a file.
d. **Map**: It defines a map and its properties, including the map name and existence status.
e. **Order**: The order class represents orders in the game, specifying source and target countries and the number of armies to be transferred.
f. **Player:** This class represents a player in the game, including their name, ID, number of armies, countries owned, orders issued, continents owned, and cards possessed.

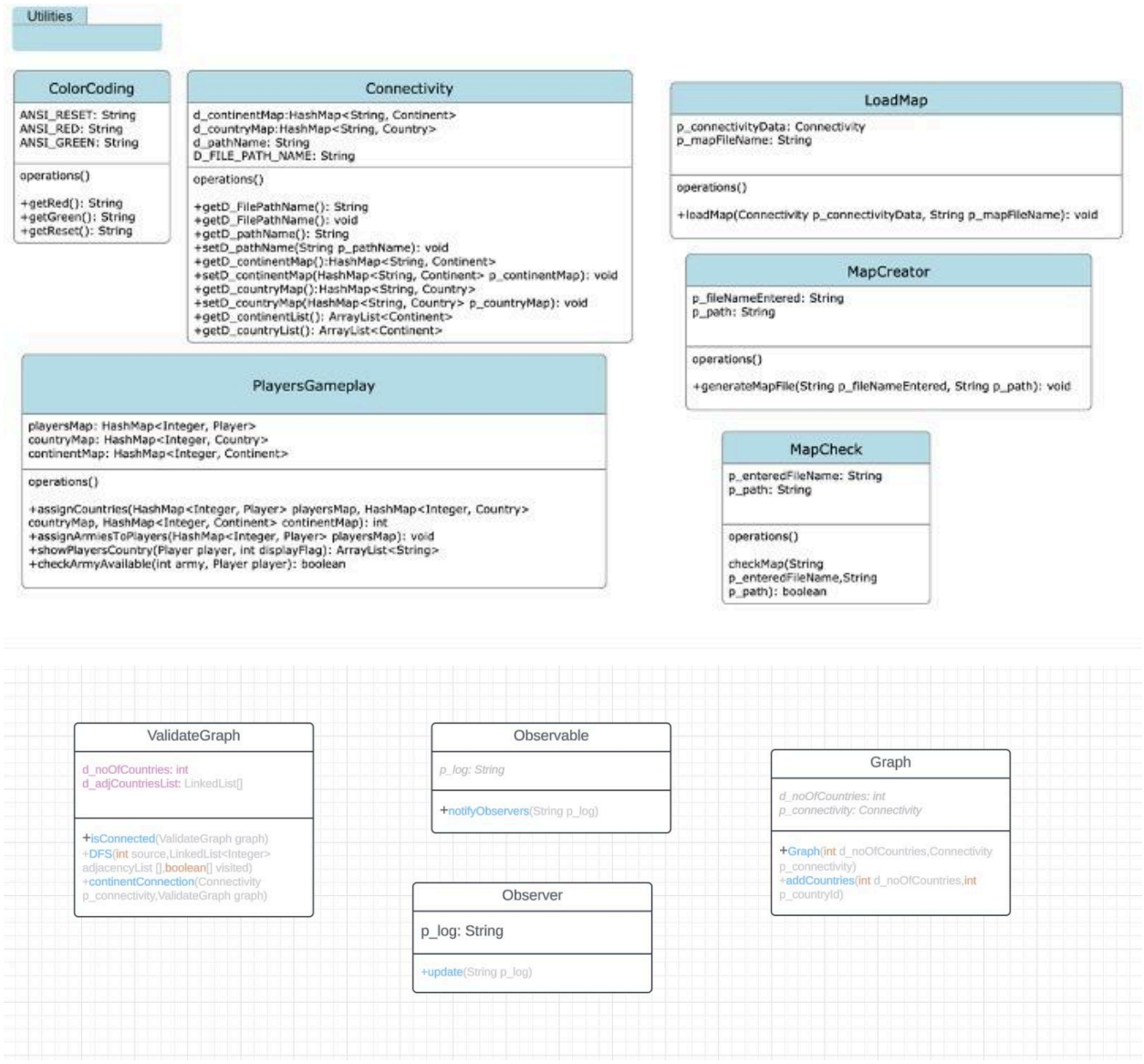
State:

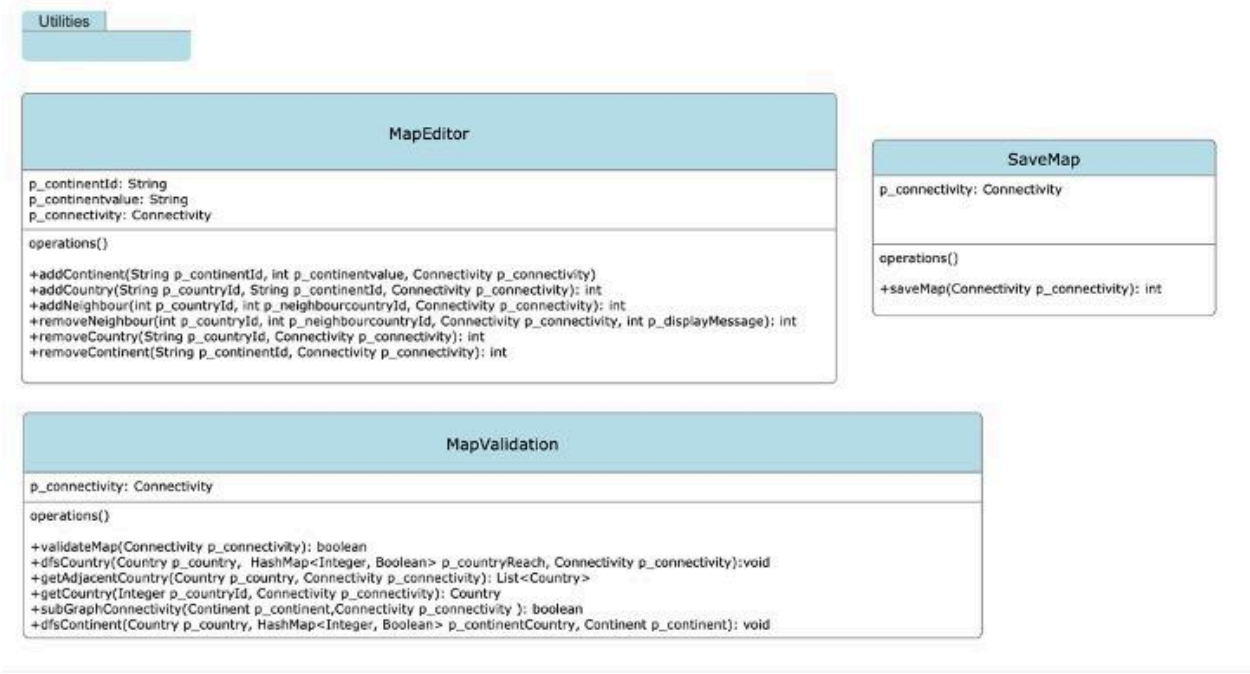


a. **Attack**: It represents the Attack phase in a game, handling player commands related to attacks.
b. **Edit**: It represents the editing phase in a game, providing methods for editing various aspects of the game map.
c. **End**: It represents the end phase of the game, handling actions and commands associated with ending the game.
d. **Fortify**: It represents the phase where players can choose to fortify their countries, allowing them to strengthen their positions on the game map.
e. **MainPlay**: MainPlay class is a ConcreteState in the StatePattern. It encapsulates the behavior for commands applicable for states like Reinforcement, Attack, Fortify while indicating invalidity for others. This state encompasses a set of states, providing shared behavior among them. All states within this set must inherit from this class.
f. **Phase**:This class manages the different states within the State Pattern in the game. It serves as a base class for various game phases, providing common functionality and defining abstract methods to be implemented by concrete phase classes.
g. **Play**: It defines the behavior for commands that are valid during gameplay phases and signifies invalid commands for other phases. Instances of this class represent a group of gameplay states and provide common behavior for all states within this group. All states within this group must extend this class.

h. **PlaySetup**: It represents the setup phase of the game where players can add or remove players, assign countries, and begin the game.
i. **PostLoad**: This class represents the phase after loading a map. Allows the user to save the map and proceed to game play phases.
j. **PreLoad**: The class represents the phase before loading a map. Allows the user to load, edit, and validate the map before entering the PostLoad phase.
k. **Reinforcement**: This class represents the game's reinforcement phase. Allows players to deploy armies and proceed to the Attack phase.

Utilities:

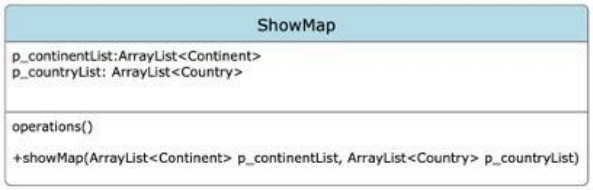


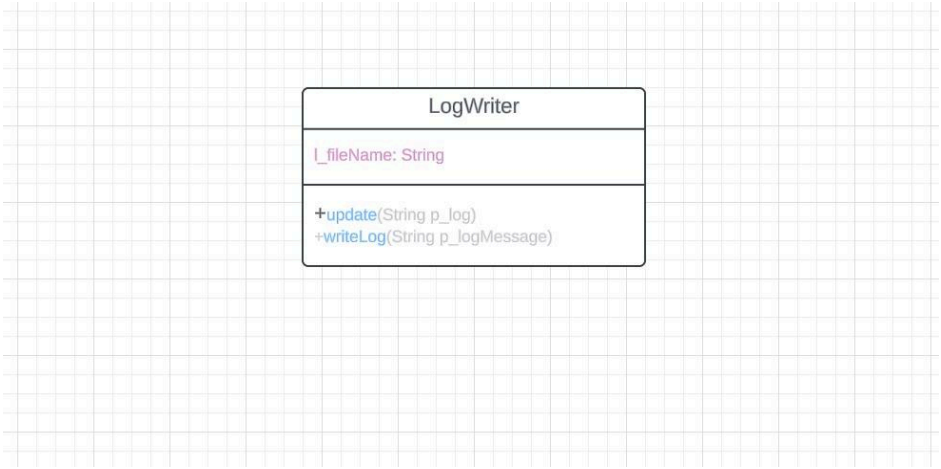


a. **Connectivity:** This class serves as a medium to transfer data from one part of the map to another and binds it. It maps continent and countries names to their objects allowing a smooth gameplay.
b. **LoadMap**: Responsible for loading the map selected by the user.
c. **MapCheck:** Responsible for checking whether the map exists in the directory path or not.
d. **MapCreator:** Responsible for automatically generating a new map when the user asks for it.
e. **MapEditor**: The MapEditor class facilitates the adding and removing of countries, continents and neighbors in an existing map.
f. **MapValidation:** It is responsible for validation of the entire map to check for correctness.
g. **PlayersGameplay:** Manages the entire gameplay of the player from assigning armies to checking its availability to deploying.
h. **SaveMap:** Responsible for saving map after the user has created a new map or modified the existing ones.
i. **ColorCoding:** It provides ANSI escape codes for displaying texts in red, green and setting it to default.
j. **Graph:** Responsible for representing a graph of countries, initializing it, and adding countries and their connections.

k. **Observer:** The Observer interface defines a standard way for objects to observe and update log messages.
l. **Observable:** interface defines a standard way for objects to notify observers about changes by using the notifyObservers() method.

View:





a. **ShowMap:** Responsible for displaying the continents, countries, army counts.
b. **LogWriter:** Observes log messages and writes them to a log file in a directory.