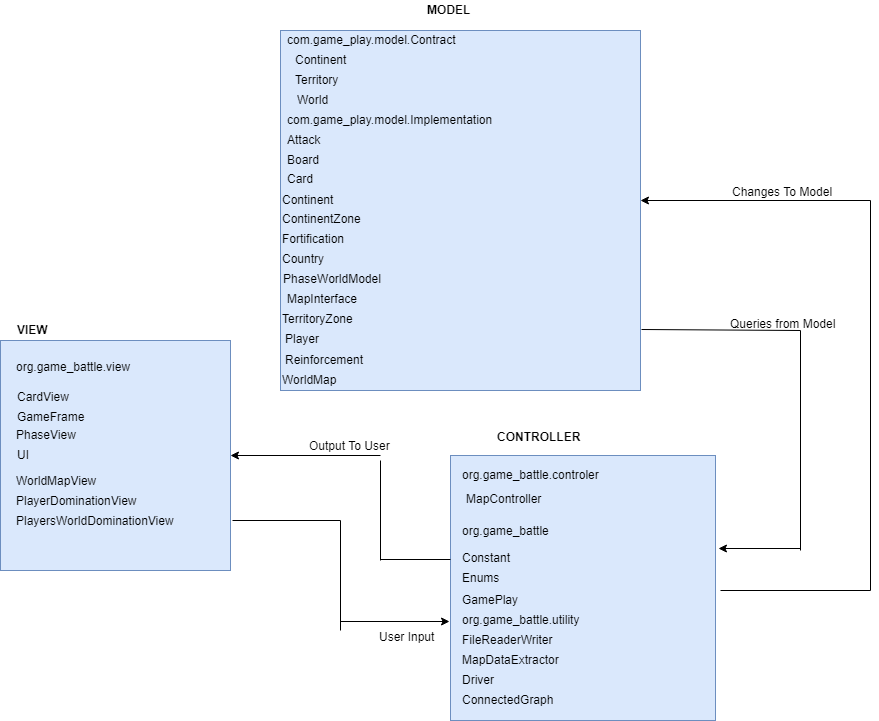
**Architectural Design:** The game which we have implemented is inspired from Model View Controller (M.V.C) architecture, but it is not a **COMPLETE** implementation of **MVC**. We have divided and distributed our modules in the parts **MODEL**, **VIEW** and **GAMEPLAY**. Below is the block diagram of our implementation of MVC architecture. All the states of the game, like player, cards, continents, countries etc. are in our model. The view is not directly interacting with the model. The logic is inside controller and gameplay folder, which can be related to controller in MVC but is not exactly the controller in **MVC.**



**Model:**

* **Player:** The reinforcement, fortification phase is implemented here and this class is used to define the players and their countries and continents**.**
* **Country:** Country class defines the set of getters and setters for armies and countries and their neighbours.
* **Continent:** Continent defines the setters and getters for continents, control value and countries belongs to a specific continent.
* **Board:** Board class defines the initialisation of the players and distribution of countries to the players.
* **Card:** Card class is used to define the set of cards which we are using in the gameplay.
* **MapInterface:** This class defines map creation, editing and connectivity of the map.
* **ContinentZone:** Class represents the continent name and it’s country list.
* **TerritoryZone:** Class has country name ,coordinates and the neighbouring countries.
* **WorldMap:** Class has set of map implementation like continent, country, neighbouring countries and store this value in a hash map.
* **Continent (Interface):** It is an interface which has a list of methods defined for the Continents.
* **World (Interface):** It is an interface which has list of methods defined for the continent, country and the neighbouring countries.
* **Territory (Interface):**It is an interface which has a list of methods defined for territories, its coordinates and the neighbouring countries.
* **PhaseWorldView:** Phase world view represents the display of changes from the model which gets updated with every state change.

**Business Logic:**

* **ConnectedGraph:** Checks whether a graph is connected or not and also validates the map.
* **Gameplay:** It has the following functionality**.**
* **ReinforcementPhase:** It has methods to get the number of armies calculated for are assignment to each player.
* **AttackPhase:** Player can declare an attack by selecting attacker and attacked country. Attacker and attacked player decide how many dice to roll. Proper number of armies are deducted from attacker/defender country during the attack(s). If defender is conquered, attacker can move any number of its armies in the conquered country. If It results in conquering the whole map, the attacker is declared the winner and the game ends. Player may decide to attack or not to attack again. If attack not possible, attack automatically ends. Implementation of an “all-out” mode, where once the attacker and attacked country have been identified, the attack proceeds with maximum number of dice and end only when either the attacker conquers the attacked, or the attacker cannot attack anymore.
* **FortificationPhase:** It provides method to pass the armies form one country to another.
* **RoundRobin:** It provides functionality for round robin traversal among the players. Controller: It has controllers for controlling the models.
* **FileReaderWriter:** Reads the already saved map or writes the newly created map.
* **MapDataExtractor:** It takes the country and continent values from user and sets the continent and country information.
* **MapController:** It describes the functionality of map editor, connectivity and load.
* **Driver:** It describes the model,view and controller operation of the map interface.

**View:**

* **UI:** Provides an interface for the user to interact with the game. It launches a pop up for player selection, cards selection, placement of armies and fortifications of armies conquered.
* **WorldMapView:**This class takes the input from the user either to create a new map or load an exiting map and an option to edit a map in the console.
* **CardView:** Displays the card view of the exchange of cards.
* **PhaseView:** Displays the phase of the game play currently happening.
* **PlayerDominationView:** Displays the view of player percentage and domination.
* **GameFrame**: Displays the UI prompt for players and cards selection.