Overview of our framework:

**view module**

This module is responsible for interacting with user and executing commands

**controller module**

This module is responsible for processing commands, game rule etc.

**model module**

This module is used for storing data objects like Player, Countries, Continents etc.

Classes in view module:

MainView - This class contains the main function and is used to start the map editing console

ValidateCommandView - This class is used to validate entered command for map editing phase. It will check if a command is valid for current phase and it will also check the syntax and number of parameters passed.

ExecuteCommandView - This class is responsible for communicating with the controller. It is responsible for calling appropriate functions from controller as per the entered command.

ConsoleView - This view is used to provide console for entire map editing phase

MapEditingCommandListForUser - This is an enum class which stores all hardcoded commands for map editing phase.

Classes in controller module:

GameEngine - This class implements functionality related to the main game engine and also implements console for initial Game play commands like loadmap, gameplayer and assigncountries.

MapController - This class is responsible for maintaining the map. It takes in command and calls appropriate setter/update functions in respective map model class.

Classes in model module:

ContinentModel - This class will store information related to specific continent. Its control value and list of country that belong to the continent.

CountryModel - This class will contain information related to specific country. Its current armies count, player who owns it and the continent it belongs to.

CoordinateModel - This model stores coordinates of the country with respect to the entire map.

GamePlayCommandListForPlayer - This is an enum class which stores hardcoded commands issued by the player during the main game play.

MapState - This class is used to maintain the list of continents and countries present in the map and their relationship.

Order - This class is used to store order issued by a player. It contains the variables to store command and its parameters. It also holds reference to the player who issued the order.

PlayerModel - This class is used to store a players’ information like countries owned and its name. It also stores temporary information like reinforcements and list of orders.

PlayerState - This class is used to maintain the state of the players like add/remove player.

Incorporation of State pattern:

Created Phase abstract class and multiple sub-classes of Phase to indicate different phases. For each command entered by the user/player, we first validate the command and call resp. methods. Then base on the Phase, the method gets executed or it prints “Invalid command for … phase” message.

Incorporation of Command pattern:

Created Order abstract class and multiple sub-classes of Order to indicate different orders. For each command entered by the user/player, we first validate the command and call resp. methods. Then base on the Order, the method create respective Order object and it gets issued.