Final Documentation

# Model Design Changes from initial UML :

## Player

Player has been replaced by an abstract class called figure, implemented by the human Player class which has a username and additional methods for picking ammo and fields for the inventory which a terminator has not, and the bot (Terminator) class which has an additional shooting and simplified canGo() method.

## ActioN

Actions have been moved out from the model into the controller as it was much easier to handle actions using controller states instead of implementing complex Action classes with an execute action method as originally intended

# Patterns choices:

## State

Controller has been implemented using a technique similar to state pattern to increase legibility of the code, each kind of player action is handled by a different controller state which calls different methods of view in different ways to take and handle input data, once a state has been executed the state is reset to a default ChoosingMove state which corresponds to the default menu during game and calls its execute method which determines the next state of the controller and again invokes the execute state method of that state.

When the game is over an EndGame state is called and executed, which will proceed to display the leaderboard to the clients.

## Observer

Observer pattern is used to sent a .json update to the views in order to update the clients during the course of the game.

## Factory

Card and Map classes are generated using builder classes which read from file to create objects, instead of implementing the overly complex parsing and file reading directly into the constructor of those classes.

## Adapter

Since the server end uses the actual model and the client end only works with strings from .json files an adapter class is used to manage the connection of the two view interfaces, since the server view interface needs model Objects and the Client only can handle strings. The ServerCommManager class is ergo needed to convert objects into .json and to parse the client’s response into Objects that can be handled by the server.