Identified Patterns W/ Description of Use

1) Observer Pattern
    a) The observer pattern is used in the form of our event bus to update the client when there have been changes to the game in the server.
    b) The “clientActivity” class subscribes the object “gridEventHandler” to handle changes to the grid from the server.
    c) In the class “Game” it posts when a new tank spawns
    d) The “gridPollerClass” posts when the grid updates or when there is a new board event
2) Singleton Pattern
    a) The class EventHistory is a singleton and doesn’t need to be multiple to keep a log of events for replay.
3) Facade Pattern
    a) The GameController class receives commands and requests from the client, it then directs the InMemoryGameRepository to execute the requests within the subsystem that is the server system. The facade is InMemoryGameRepsitory, the subsystem is the server classes (Tank, Game, Bullet).
4) Builder Pattern
    a) The Game Board will need to be extracted into its own class so that a builder can be called from the FieldHolders class which will then build the game board based on certain input from FieldHodlers.
5) MVC Pattern
    a) The MVC pattern is used in the client to separate what classes are responsible for each part of the MVC
    b) The ClientActivity class and other parts of the util folder in the client are the controllers responsible for instructions that will change the model.
    c) The gridAdapter class is responsible for changes to the view.
    d) The facade pattern in the server fulfills the model role by taking commands from the client’s utils and executing them to update the board.