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 to the object "gridEventHandler" to handle changes to the grid from the server.
        i) In this case the gridEventHandler is the subject and clientActivity is one of the observers
    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
    e) The gridPollerTask is another subject with its observers being gridUpdateEvent and the BulletZoneRestClient which updates the server-side.
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.
    b) The client of EventHistory is the gameStateController which makes calls to the singleton state of eventHistory.
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 class holds a reference of the GameBoard. The inGameMemoryRepository knows about the GameBoardBuilder and is its director. So when a GameBoard must be made, it is called from the inGameMemoryRepository class and the reference in Game is updated.
5) Command Pattern
    a) On the server side events like move, fire, and turn are checked and completed and then stored in the gameBoard which has a reference inGameMemoryRepository. Rather than all events being executed and sent to the client individually they are stored in the repository and then accessed by the client all together. Then on the client side these actions are expressed from the received board through gameProcessor and gridAdapter.
6) 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.
e) The TankEventController controls updates to tanks and other game events. Changes here are expressed in the view by the gameEventProcessor which also holds model data in the form of an array of entities.