Progetto di Fasanella, Maggioni e Manini

Sequence diagrams and description of the network handling

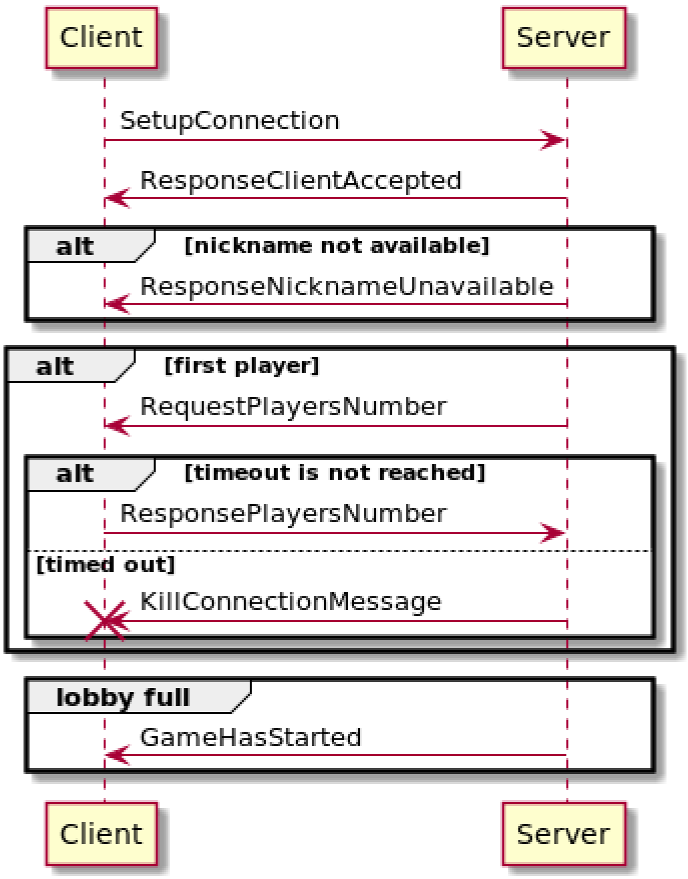
High-Level description for Client connection

The server supports simultaneous games, it’s composed of a single lobby where players are accumulated until it’s full and a register to associate client and Game.
For each game, a Game class is created which will instantiate Controller and Model and will have access to the VirtualClient s in that game.
Each VirtualClient contains information about the client socket and exposes methods to make requests to the view throughout the network (which remains invisible to the Game class itself).

Low-level description for Client-Server communication

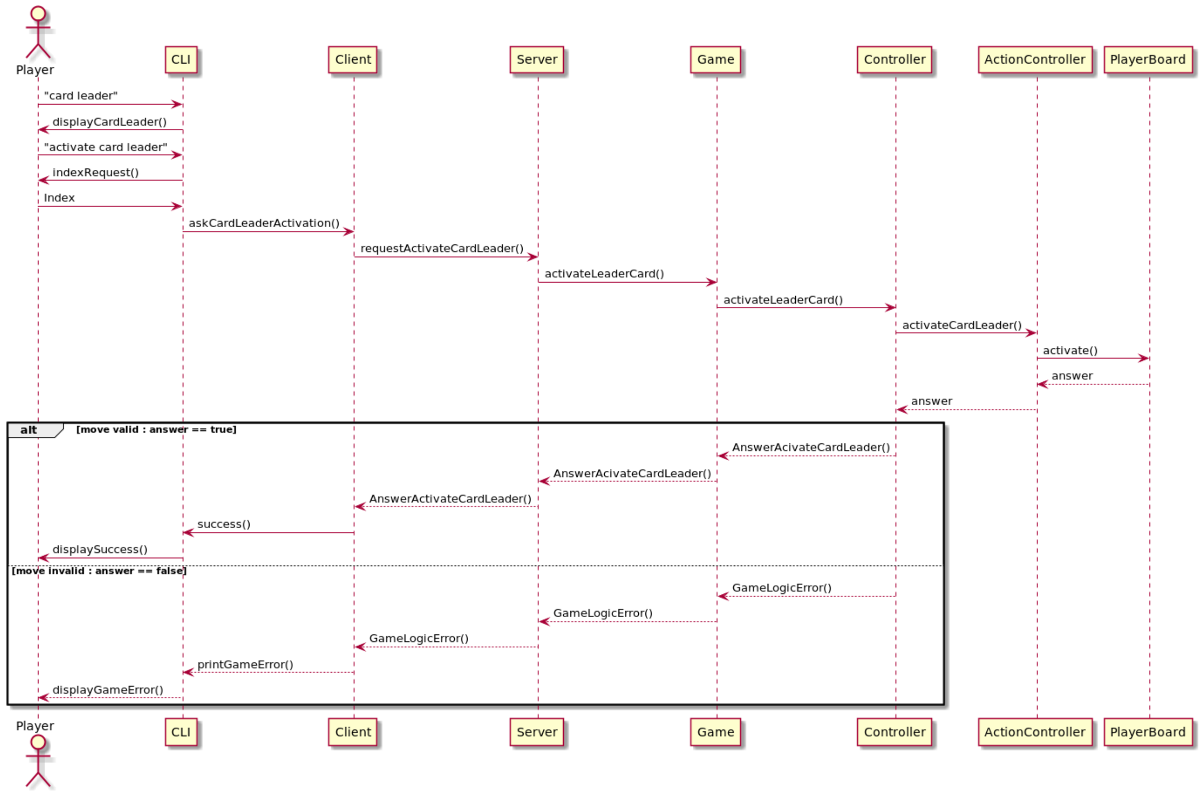
In the connection, serialized classes are exchanged using sockets and the libraries java.io.ObjectOutputStream e java.io.ObjectInputStream.
All these classes will extend SerializedNetworkMessage which will then be divided into 2 main classes (ClientMessage and ServerMessage) and subsequently (ServerRequest, ServerResponse, ClientRequest, ClientResponse) and themselves extended as needed from Server, Controller, and View.
These classes will implement a Strategy Pattern and calling their activate() method will trigger a method in either VirtualClientCommandDispatcher or ClientCommandDispatcher.
If the message extends a Server or Client Request, the associated TimeoutID is sent back with the Response deactivating the timeout. This allows us to use synchronous requests throughout the Network avoiding use of the Observer Pattern in most cases.
The TimeoutHandler class exposes the method sendAndWait() which stops the caller from executing until a response for his message has been processed by the receiving thread in the given time.

Connection Player Server



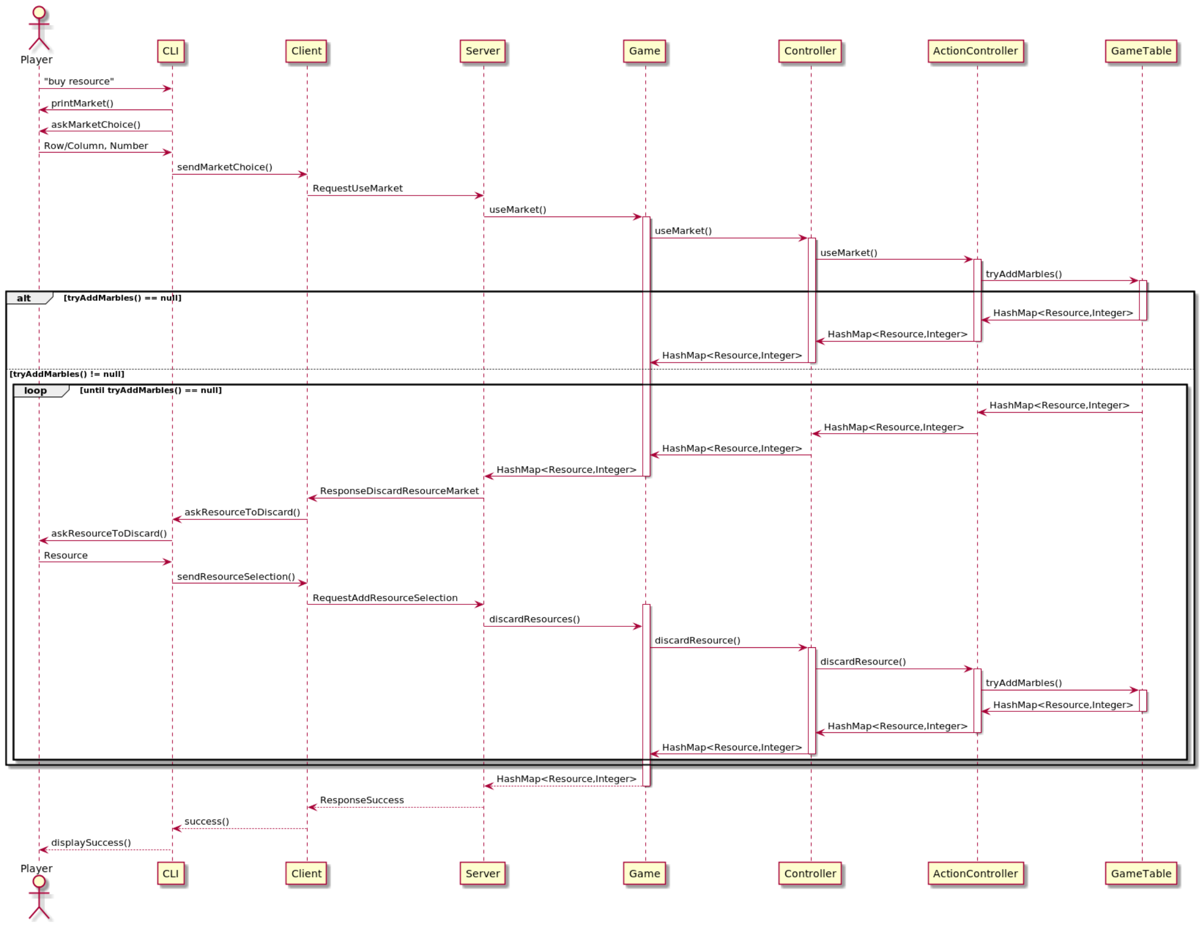
The Client, after verifying the Server's existence, sends a serialized class SetupConnection(). Depending on the number of players connected to the lobby, the Client may be asked to set the number of players on the game. Otherwise he gets a response detailing the number of players connected and the remaining slots. When the lobby is full, messages are sent to the players to inform them of the beginning of the game.

Card Leader activation



The client sends a serialized class RequestActivateCardLeader and the server answers either with a high-level class GameLogicError, or a ResponseSuccess class containing a successful message. Further communications between Client and Server regarding cards leader are handled in the same way since those classes only allow to be used via the activate() method, which can be called repeatedly if needed (e.g. tho cards with conflicting effects for example if the same type).

Market usage



The client sends a serialized class RequestUseMarket in which details regarding which row and column are selected. Depending on the resources withdrawn, the client might be asked to select resources to discard from the ones taken, one or more at the same time, until the addiction to the deposit would result in a consistent deposit state. At this point, a successful message class is sent to the player