My Shelfie: Comunication Protocol

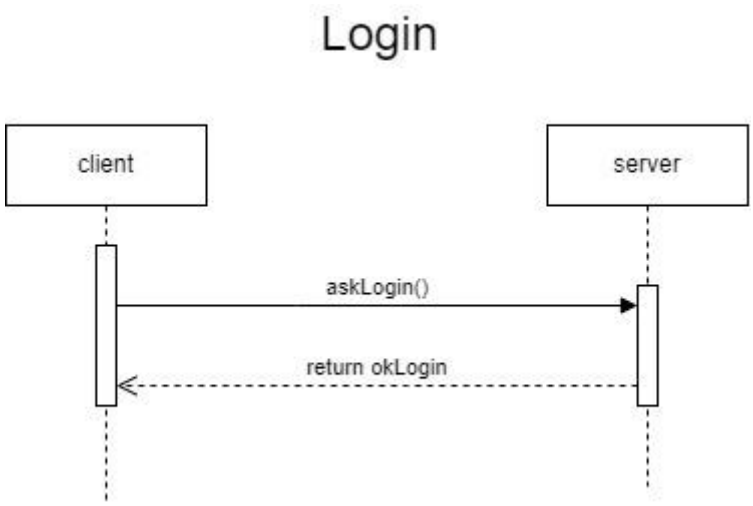
IS23-AM05 Calvi Mirko, Calzolaro Simone, Caratti Elena, Clara di Gioacchino Gabriele

General info:

Due to the structure of our network implementation, the RMI and Socket versions of the communication protocol override the same methods of a more generic superclass. The aim of this implementation is to guarantee a usage completely independent from the type of communication used. For example, if the client needs to send a message it will call a generic “Client.askPostMessage()”, depending on the dynamic type of the class it will be called “RMIClient. askPostMessage()” or “SocketClient. askPostMessage()”. The same logic applies to the server when interacting with a client.

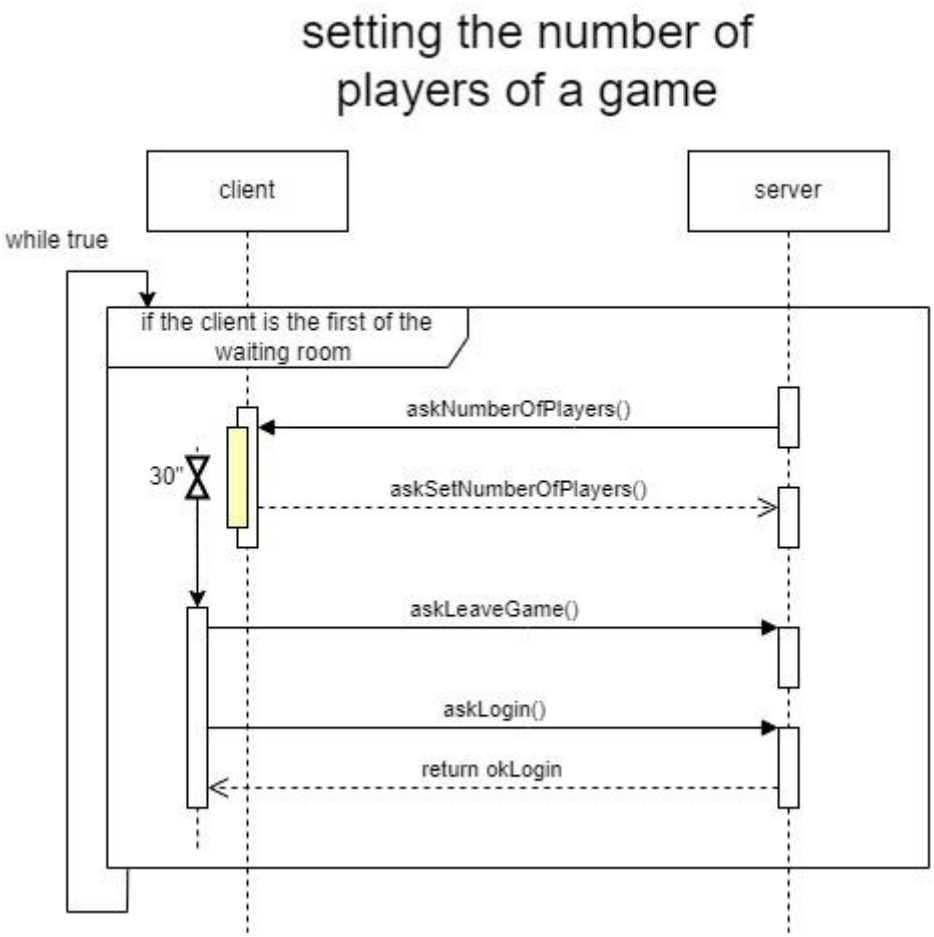
All the next cases apply to the described methodology. The methods written on the arrows change their implementation depending on the type of connection.

LOGIN



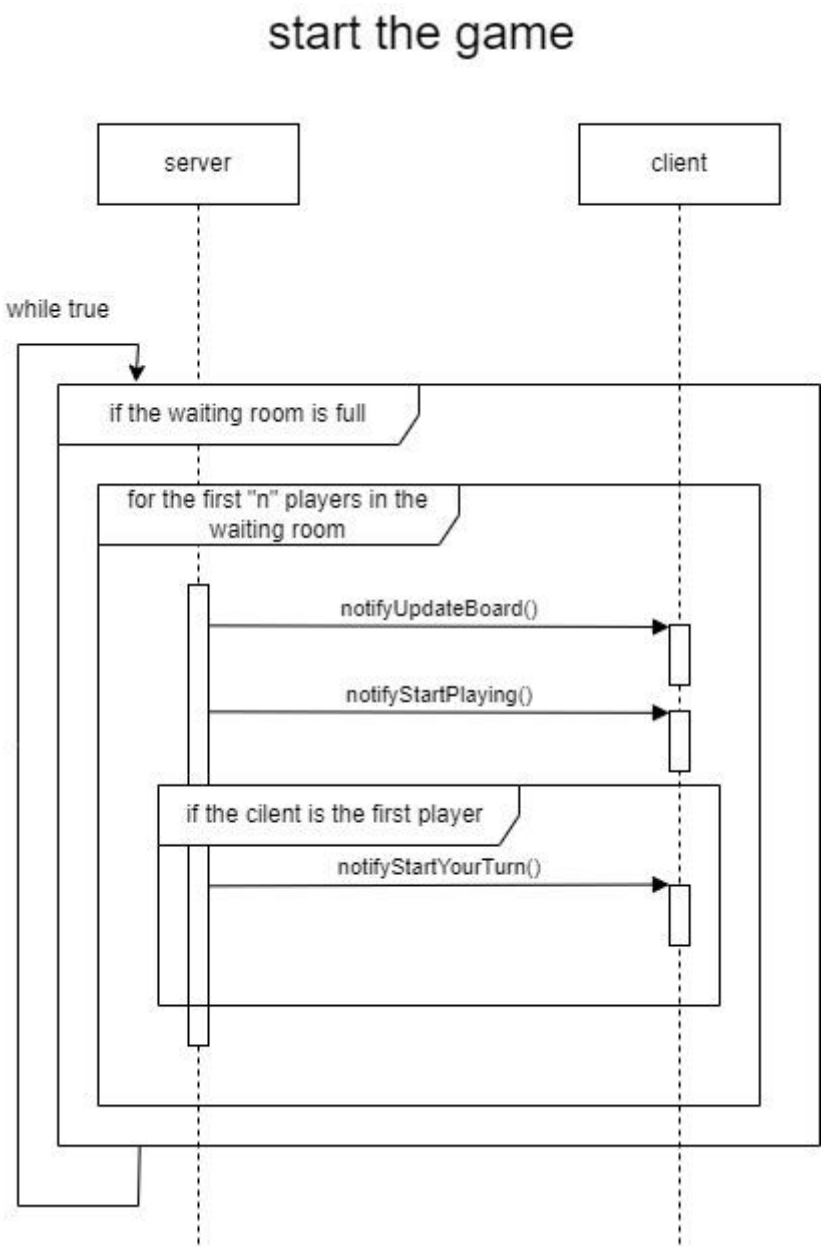
Ask the server to log in the waiting room. The server check that doesn’t exists any player in the waiting room with the same nickname and, if true, adds the player, else throws an exception.

SET NUMBER OF PLAYERS



The waiting room works as a FIFO. If the waiting room is not empty and if the “oldest” player in the waiting room hasn’t been asked the number of players he wants in his game the server calls askSetNumberOfPlayers(). The user have max 30 seconds to insert and send to the server the chosen number, otherwise it leaves the game, he is deleted from the waiting room and the login process restart. If the given number of players is permissible the server accept it and wait for the waiting toom to fill.

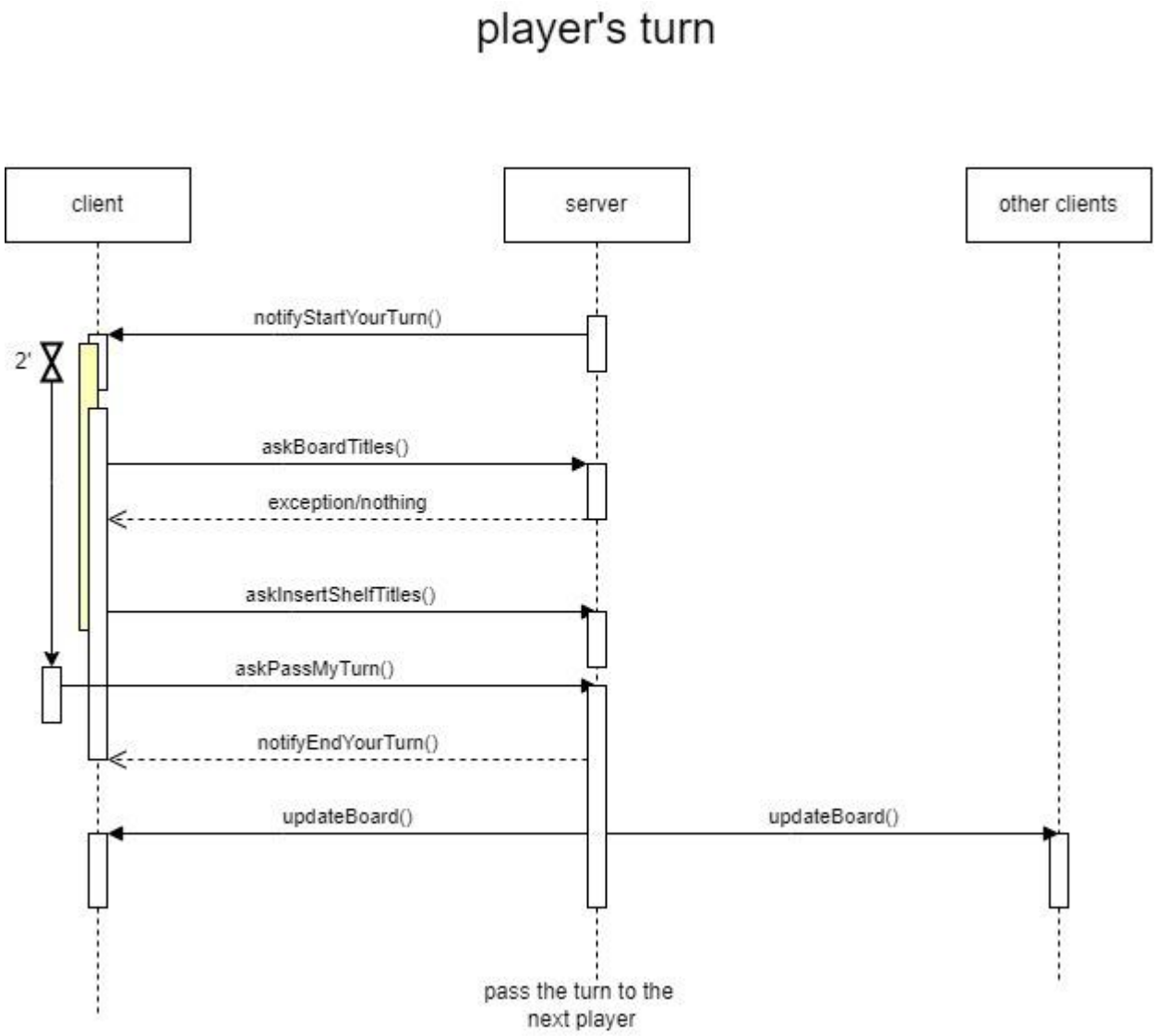
STARTING A GAME



When the waiting room reach its full capacity (see paragraph before this) all the first “n” clients are notified, where “n” is a number between 2 and 4. The server with “notifyUpdateBoard()” push to the client the Board of his game, his bookshelf, a map containing other players’ nickname and bookshelf and the current score of the player. After that the server using “notifyStartPlayng()” tells the client that the game has started and reports all the common Goal Cards, the Personal Goal Card and the ID of the game he is playing.

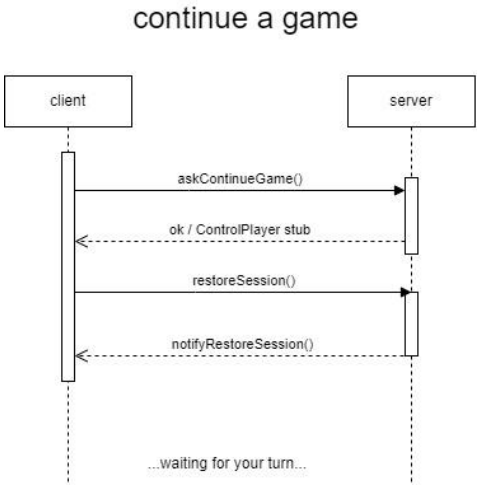
If the player notified is the first, he is also notified to start his turn.

GAME TURN



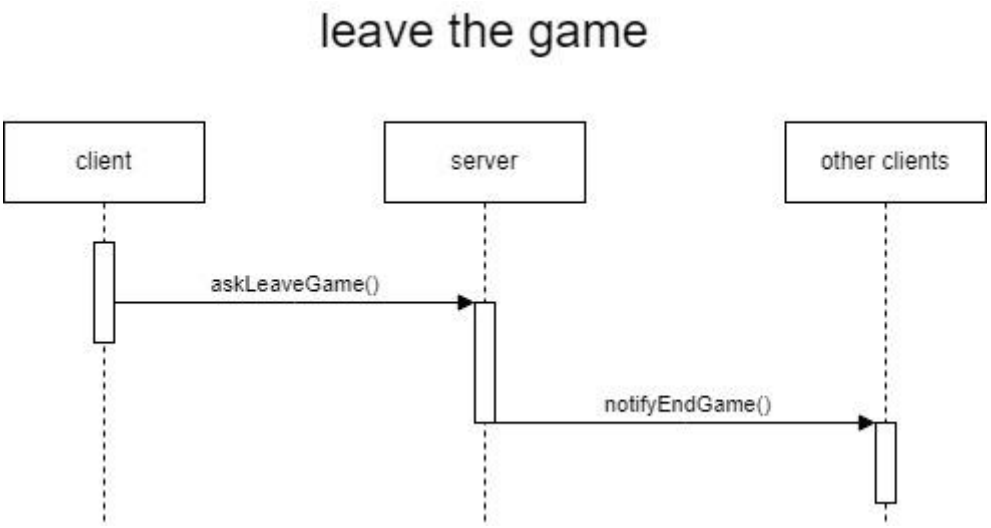
After the server notify a client to start his turn the user pick some tiles from the board and asks the server if the chosen tiles are valid, then the user chose a column of his board where to insert the new tiles and tells server of his modifications. If the modifications are valid the server-side model is update and all the clients are notified.

CONTINUE GAME



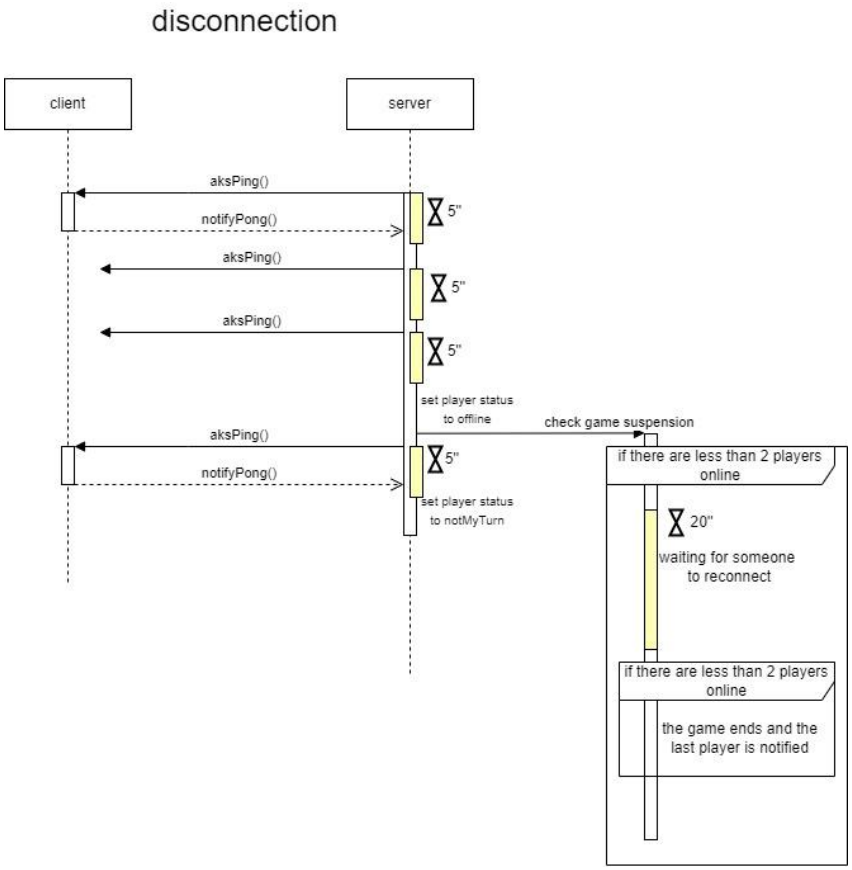
The askContinueGame() is called when after a client-side crash the client app reads a JSON file and find out the user was playing a game, so asks the server to come back in his game. If the game is still going on, the server accept his request and updated him with the current state of the board and other players bookshelf

CLIENT LEAVES THE GAME



When a client asks the server to leave the game he's playing and the server accept the request, all the players are notified with the rankinig of the match and the game is deleted from the lobby.

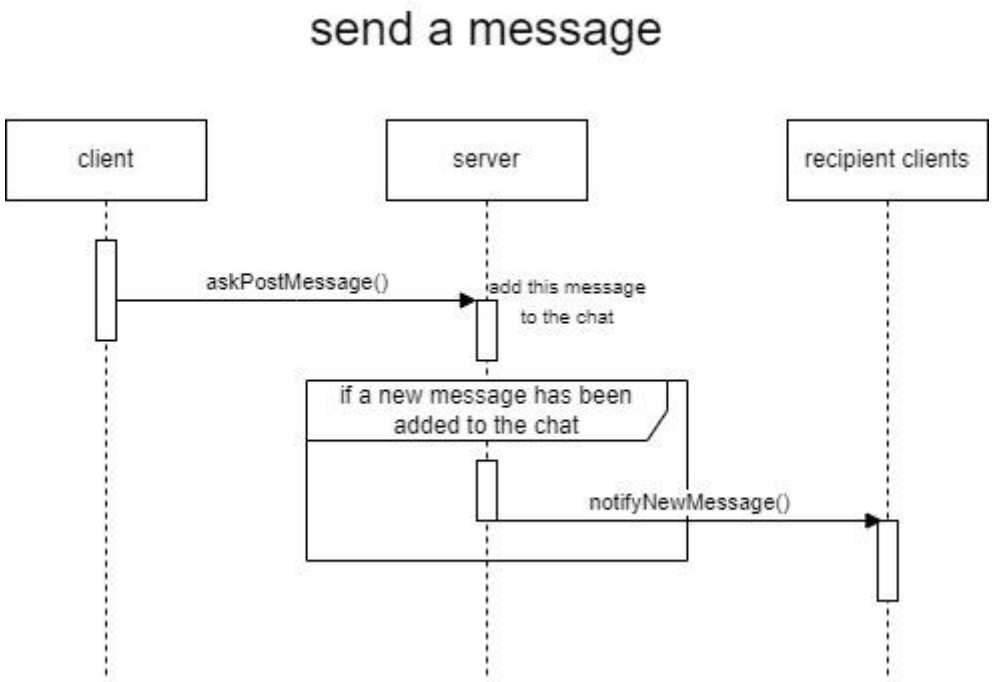
DISCONNECTION OF A CLIENT



The server ping the client and waits 5 seconds , if the client has sent back a response it means he is online, otherwise, at the second ping fail the status of the player is set to “offline” and he can’t play until is online again.

In the worst case scenario where all the players are offline except one, the “check game suspension” condition is verified and, if after 20 seconds no one has rejoin the game, the match ends.

MESSAGING PROTOCOL



After a client send a message containing the text message and its recipients the server adds the new message to a list containing all the messages and their senders. Meanwhile, another process notices the addition of a new message to the conversation and notify the relatives addressees.