Sequence Diagrams Description

Flow-Player joins a lobby-match

1) The CLI/GUI asks to join a Lobby, specifying what Lobby. This is specified to the NetworkClientMatchController
2) The NetworkClientMatchController forwards the request to the UserOfClient, invoking the sendAndWaitMultiple: this method is used to send a Message to the Server and busy-wait till an answer Message is received; the answer Message must match one of the ones waited
3) The UserOfClient forwards the request to the Client class: this is the class that actually sends the Message using the network

4) The Message is received by the ClientController for that Client on the Server
5) The ClientController forwards the received Message to the Server class
6) The Server forwards the received Message to the ListenLoop for dispatching
7) The ListenLoop invokes react(Message) to fire up the correct reaction to the received Message
8) In this case, the reaction is taken care by the MatchController: it will basically try to perform the request action (lobby joining) and answer back to the Client with a Boolean flag indicating whether the operation was successful or not
    a. In case of success, the MatchController will
        i. Set the State of the Client in the Server to “InGame”
        ii. Ask the GameController to reconnect the player to the Match (so it is able to start playing where it left (from scratch or from a previous state if the player previously disconnected while playing)
        iii. Fire up the MATCH_COMPOSITION_CHANGE_EVENT handler that will basically inform all the connected player to the Match about the new player in the Match; the “inform process” is about the Server sending a Message to the Client(s) so to cause they proper reaction
    b. In case of failure, the MatchController will just inform the Client

Based on the response received, the Client will put itself in “InGame” state if the join was successful, otherwise won’t change its state.

Flow-Player place a card from his hand

1) The CLI/GUI asks to place a Card specifying what Card and the ManuscriptPosition. This is specified to the NetworkClientGameController
2) The NetworkClientGameController forwards the request to the UserOfClient, invoking the sendAndWaitMultiple: this method is used to send a Message to the Server and busy-wait till an answer Message is received; the answer Message must match one of the ones waited
3) The UserOfClient forwards the request to the Client class: this is the class that actually sends the Message using the network

4) The Message is received by the ClientController for that Client on the Server
5) The ClientController forwards the received Message to the Server class
6) The Server forwards the received Message to the ListenLoop for dispatching
7) The ListenLoop invokes react(Message) to fire up the correct reaction to the received Message
8) In this case, the reaction is taken care by the GameController: it will basically try to perform the request action (card placing) and answer back to the Client with a Boolean flag indicating whether the operation was successful or not
    a. In case of success, the GameController will fire up the PLACE_CARD_EVENT handler that will basically inform all the connected player to the Match about the placing event; the "inform process" is about the Server sending a Message to the Client(s) so to cause they proper reaction
    b. In case of failure, the GameController will just inform the Client

Flow-Player draws a card from the board

1) The CLI/GUI asks to draw a Card specifying the CardType. This is specified to the NetworkClientGameController
2) The NetworkClientGameController forwards the request to the UserOfClient, invoking the sendAndWaitMultiple: this method is used to send a Message to the Server and busy-wait till an answer Message is received; the answer Message must match one of the ones waited
3) The UserOfClient forwards the request to the Client class: this is the class that actually sends the Message using the network

4) The Message is received by the ClientController for that Client on the Server
5) The ClientController forwards the received Message to the Server class
6) The Server forwards the received Message to the ListenLoop for dispatching
7) The ListenLoop invokes react(Message) to fire up the correct reaction to the received Message
8) In this case, the reaction is taken care by the GameController: it will basically try to perform the request action (card drawing) and answer back to the Client with a Boolean flag indicating whether the operation was successful or not
    a. In case of success, the answer will also contain the Card just drawn + the GameController will fire up the DRAW_CARD_EVENT handler that will basically inform all the connected player to the Match about the drawing event; the "inform process" is about the Server sending a Message to the Client(s) so to cause they proper reaction
    b. In case of failure, the GameController will just inform the Client