Group AM64: Arbasino-Azzarà-Bianco

NETWORK COMMUNICATION PROTOCOL

The communication protocol we decided to implement consists in serialized Java objects encapsulated in a specific Java Class we called “Message”.

The Message class has various subclasses, each one has a different purpose and it is recognized through the “instance of” method, so that it can be correctly interpreted.

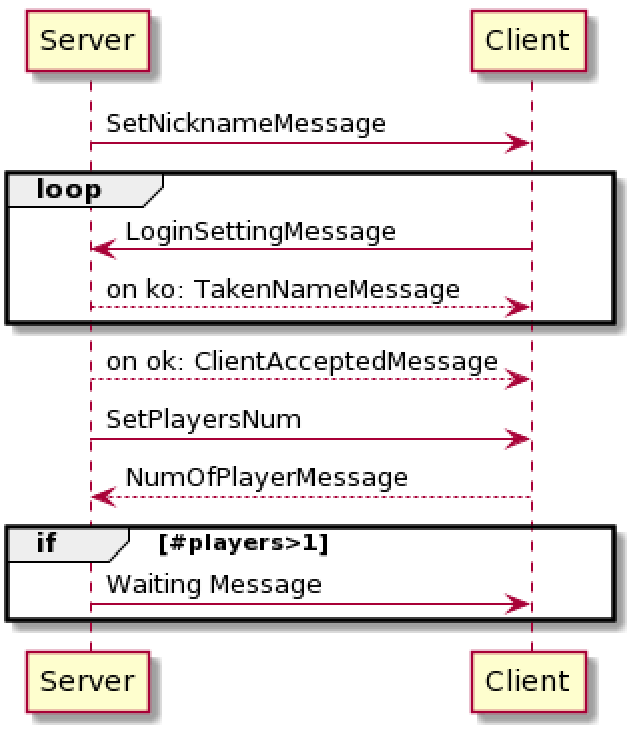
Our Client-Server communication is mostly asynchronous, as a matter of fact even if we have a StartTurnMessage the client can execute an action whenever they want, if the action is illicit a WrongTurnMessage will be sent to the them. The architecture can easily implement in the future the asynchronous exchange of messages such as a real time chat.

The server allows multiple matches and the possibility of quitting and re-joining a match thanks to the PlayerIsQuittingMessage, PlayerIsRejoiningMessage and RemoveClientForErrors.

A WinnerMessage will be broadcasted to all the players of the lobby to let them know that the match is ended and who the winner is.

For further information about the structure and the purpose of each message we advise you to read the detailed description enclosed to our Java Project.

LOGIN PHASE

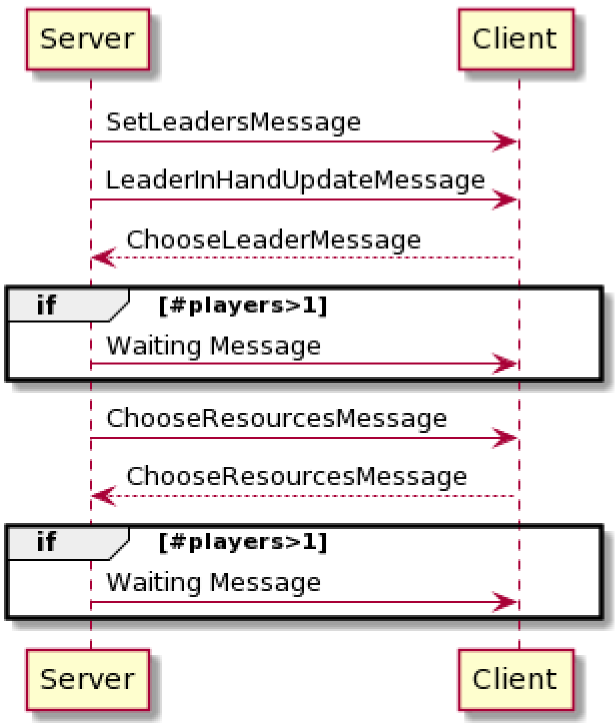


During the login phase the client is asked to type their nickname, if they want to join or create a lobby and in the latter case, they must choose how many players can be involved in the match.

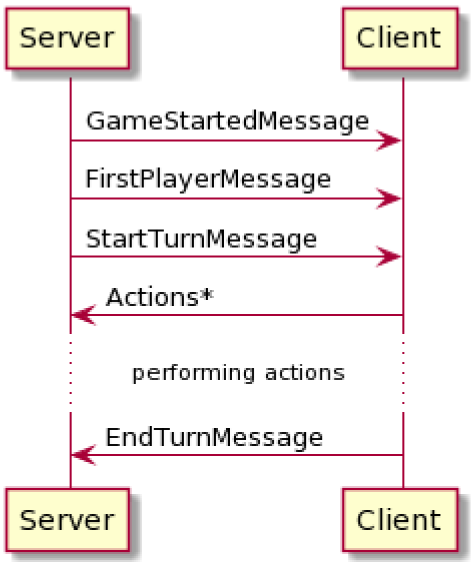
Since the nickname is univocal, the server checks that the nickname chosen isn’t already taken by another player and if so, asks for another one.

INITIAL PHASE

After setting up the lobby, the player is asked to choose two Leader Cards to discard and, according to the rules of the game, they will choose some resources (from none up to two).



GAME START



Once all the players are done choosing the cards to keep and the resources to store, the match can actually start. This is notified through the GameStartedMessage, then the FirstPlayerMessage will inform the players who goes first. The first player will also receive the StartTurnMessage.

In this phase of the game, the player can perform one (and only one!) of the main Actions such as:

-Market Action

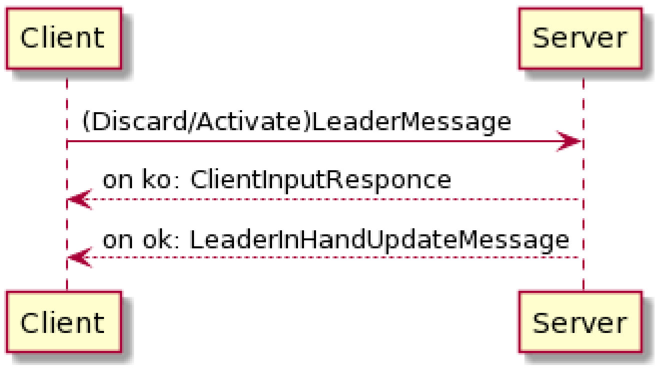
- Buy Card Action

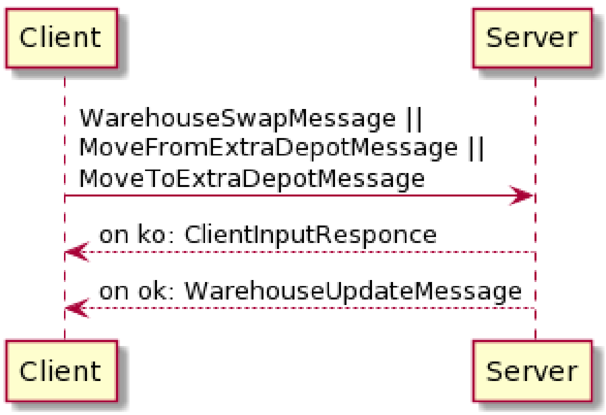
-Production Action

During the turn they may also perform as many times as they want the Swap Action and the Discard/Activate Leader Action.

The turn will end as soon as the player sends the EndTurnMessage.

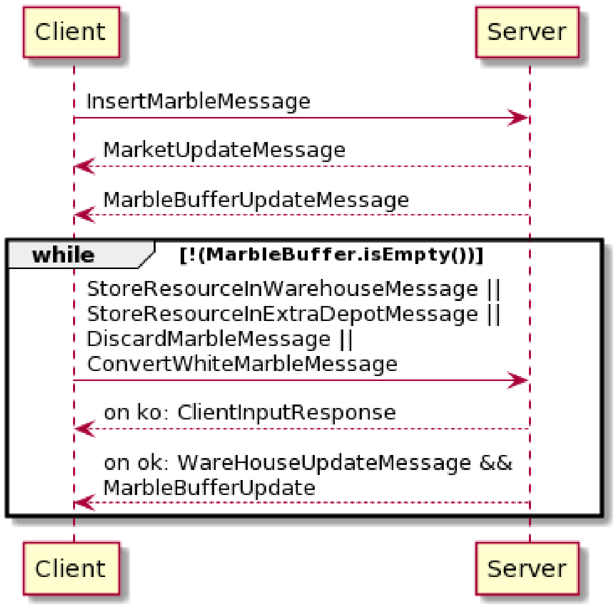
LEADER AND SWAP ACTION


Client
Server
(Discard/Activate)LeaderMessage
on ko: ClientInputResponce
on ok: LeaderInHandUpdateMessage
Client
Server


Client
Server
WarehouseSwapMessage ||
MoveFromExtraDepotMessage ||
MoveToExtraDepotMessage
on ko: ClientInputResponce
on ok: WarehouseUpdateMessage
Client
Server

As said before, these actions can be performed multiple times in a single turn. The exchange of messages in these phases consist in the communication of the will of the players to execute a certain action, then the server will evaluate the request.

MARKET ACTION


Client
Server
InsertMarbleMessage
MarketUpdateMessage
MarbleBufferUpdateMessage
while
[!(MarbleBuffer.isEmpty())]
StoreResourceInWarehouseMessage ||
StoreResourceInExtraDepotMessage ||
DiscardMarbleMessage ||
ConvertWhiteMarbleMessage
on ko: ClientInputResponse
on ok: WareHouseUpdateMessage &&
MarbleBufferUpdate
Client
Server

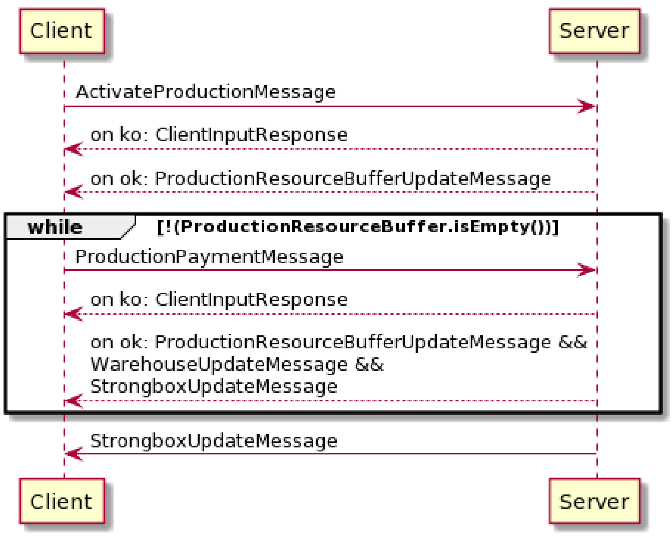
The client will choose a row or column of the Marble Market through the InsertMarbleMessage. A list of the selected row/column of marbles will be sent to the player. The action can end only if the buffer is completely emptied (the correctness of the actions on the single marbles will be evaluated by the server).

PRODUCTION ACTION

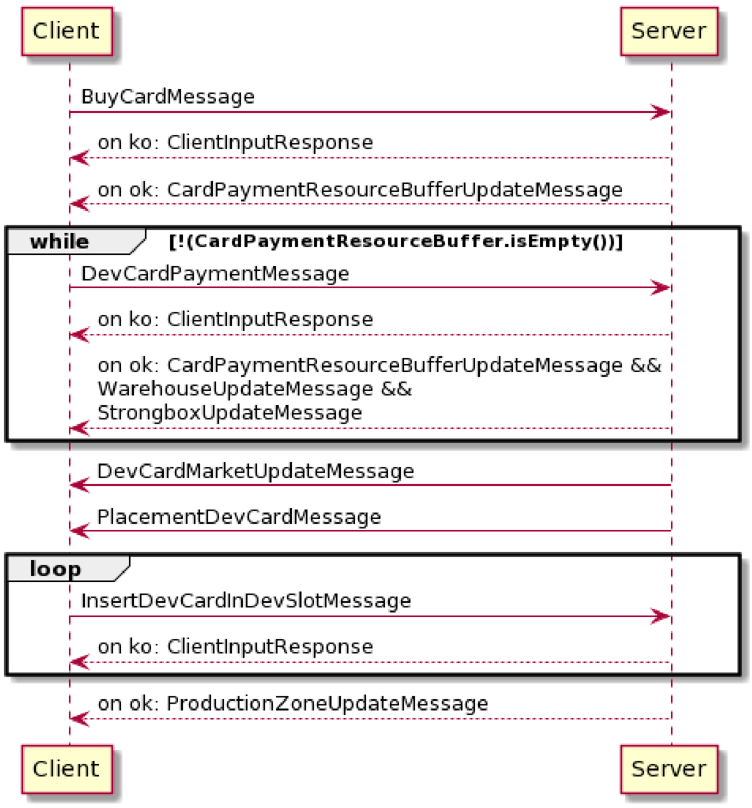
Through the ActivateProductionMessage the player chooses the combo of production slots that they want to use. The server will evaluate if they actually have enough resources to do that.

The player will then pay the resources needed.

At the end the outcome will be added to the strongbox.


Client
Server
ActivateProductionMessage
on ko: ClientInputResponse
on ok: ProductionResourceBufferUpdateMessage
while
[!(ProductionResourceBuffer.isEmpty())]
ProductionPaymentMessage
on ko: ClientInputResponse
on ok: ProductionResourceBufferUpdateMessage &&
WarehouseUpdateMessage &&
StrongboxUpdateMessage
StrongboxUpdateMessage
Client
Server

BUY CARD ACTION


Client
Server
BuyCardMessage
on ko: ClientInputResponse
on ok: CardPaymentResourceBufferUpdateMessage
while [!(CardPaymentResourceBuffer.isEmpty())]
DevCardPaymentMessage
on ko: ClientInputResponse
on ok: CardPaymentResourceBufferUpdateMessage && WarehouseUpdateMessage && StrongboxUpdateMessage
DevCardMarketUpdateMessage
PlacementDevCardMessage
loop
InsertDevCardInDevSlotMessage
on ko: ClientInputResponse
on ok: ProductionZoneUpdateMessage
Client
Server

With the BuyCardMessage the player chooses the index of the card he wants to buy. The server will check if they can actually afford it, then proceed with the payment phase.

Once the card is bought, the player must choose the slot where they want to place the card and if the chosen slot is correct according to the game rules, the action ends.