sd enterGame

Client

Server

1: Ping()

Ack()

The Ping message is used to inform the Server of the effective presence of
the client. The "acknowledgment message" is used to inform the client whether it has successfully established a connection or not.

loop : ifFailed [Guard]

2: SetNickname()

2.1: nameVerify()

Using the setNickname method, the client sends its nickname to the server, which will verify (nameVerify())if this nickname has been used by other players in the lobby or not. . Following this, an alternative condition occurs (alt): if the nickname is invalid, the client can reattempt setting its nickname using a while loop; otherwise, the server will send a message to notify the client that the nickname has been successfully added to the list of players.
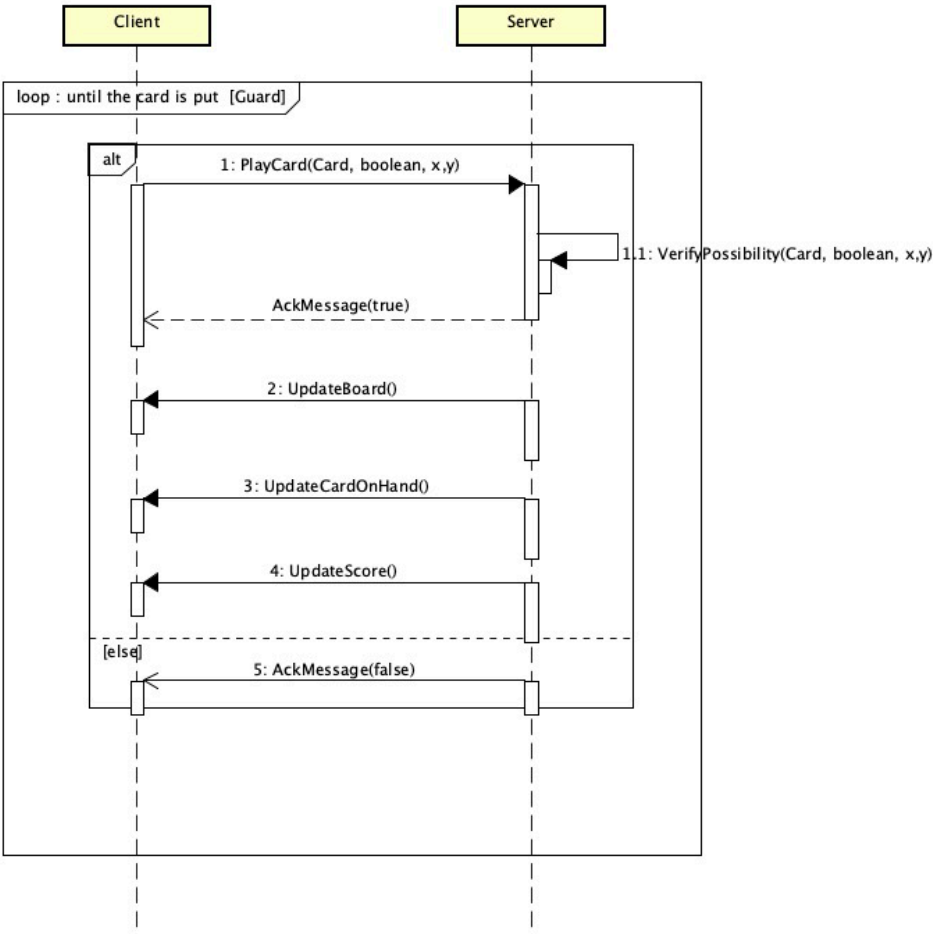
alt

NotValidNickname

[Else]

3: SetNickname()

validNickname

4: MatchList()

The server sends a MatchList to the client so that the latter can choose which game to join.

alt [if Client want a new match or there is no match]

5: createNewMatch()()

Thanks to the alternative condition, the client can choose between creating a new lobby by requesting createNewMatch() to the server or joining an existing lobby with joinMatch(). If there is no existing lobby, the client must create a new lobby.

[join a already created match]

6: joinMatch()

loop : Until all the players are ready

7: playerReadyToStart()

The client signals its readiness to start a game by invoking the method playerReadyToStart() on the server. The server can respond in three different ways:

1.If all players are ready, the server will send an AllReadyMessage() to the client.
2.If a new player joins the lobby while the process is ongoing, the server will inform the client using the method newPlayerInMatchMessage().
3.If any member of the group is not prepared, the server will notify the client by sending a message using notReady().

alt [ready]

allReady

[new Player In]

8: newPlayerInMatchMessage()

The while loop will continue until all players are ready.

[notReady]

9: notReady()

sd Sequence Diagram0

Client   Server

loop : until the card is put [Guard]

alt

1: PlayCard(Card, boolean, x,y)

1.1: VerifyPossibility(Card, boolean, x,y)

AckMessage(true)

2: UpdateBoard()

3: UpdateCardOnHand()

4: UpdateScore()

[else]

5: AckMessage(false)

The playCard method has four arguments :a card, a
boolean value, x and y coordinates. The boolean
indicates whether the card is facing front or back, and
the coordinates indicate the card's position on the
board. Using this method, the client requests the
server's permission to put a card on the board. The
server can respond in two ways:

1. If the server replies with AckMessage(true), it means
the card has been successfully placed on the board,
satisfying the card's requirements and the rules for card
placement. Following this, the server updates the board
using UpdateBoard(), the player's hand cards with
UpdateCardOnHand(), and the player's score with
UpdateScore(). Finally, the while loop ends.

2.If the server responds with AckMessage(false), it
indicates that it was unable to place the card on the
board. In this case, the client needs to repeat the
process within the while loop.

Client | **Server**

loop : IsYourTurn [isYourTurn]

1: ScegliCarta(Deck)()
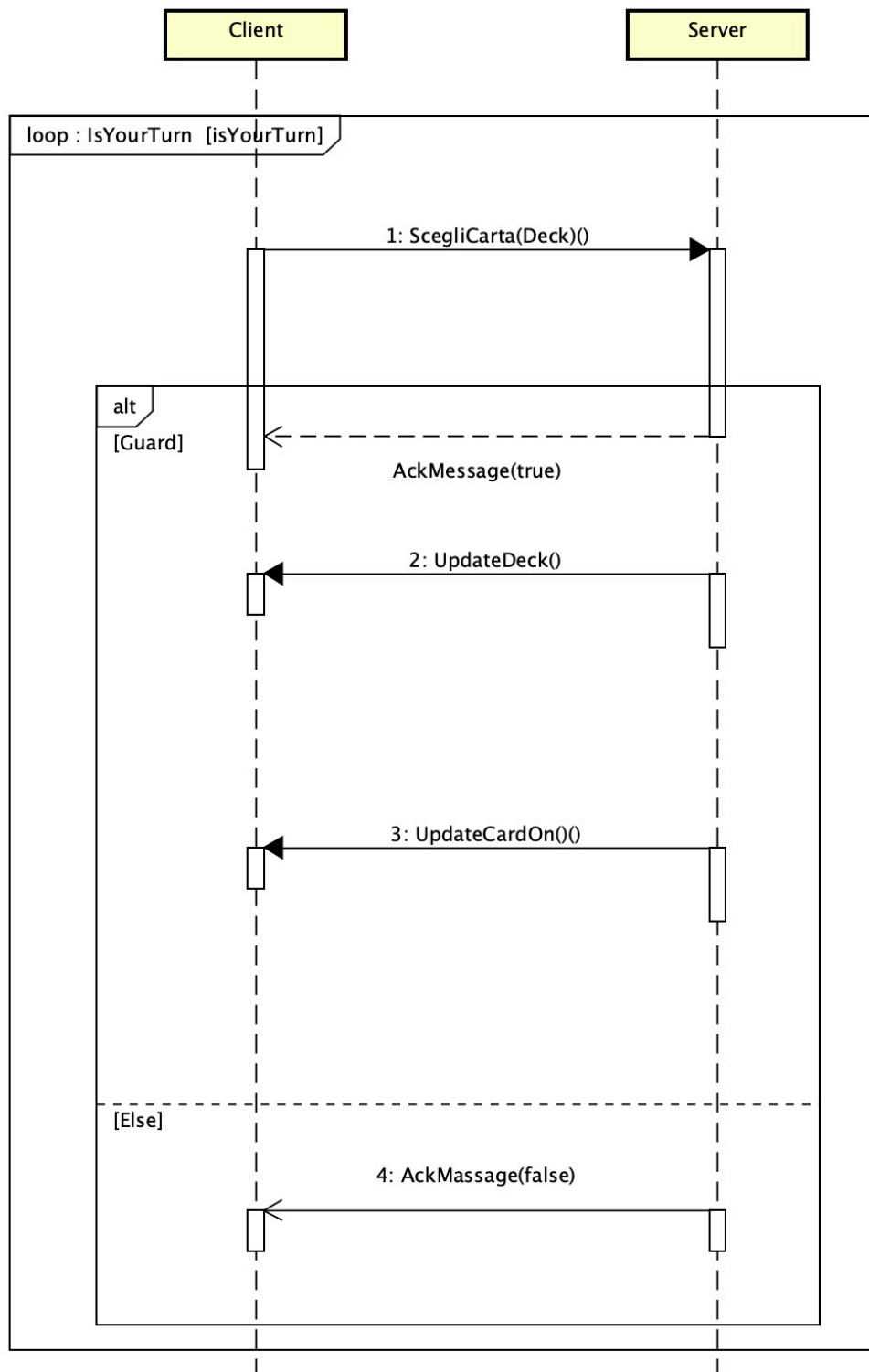
Using the "chooseCard" method, the client will request a permission from the server to select either one of the first three cards from the Resource Deck or one of the first three cards from the GoldDeck. It is always allowed to reveal the first two cards of the decks.

alt
[Guard]

AckMessage(true)

2: UpdateDeck()

If both the ResourceDeck and the GoldDeck are not empty, the server will send an AckMessage to the client, indicating that the request has been accepted. Afterwards, it will update the decks and the player's hand cards using the UpdateDeck and UpdateCardOnHand methods.

3: UpdateCardOn()()

If both decks are empty, the server will send an AckMessage notifying the client that the request has been denied.

[Else]

4: AckMassage(false)