sd game access

```
                    ┌──────────┐                      ┌──────────┐
                    │  Client  │                      │  Server  │
                    └────┬─────┘                      └────┬─────┘
                         │        connectionRequest()      │
                         │────────────────────────────────>│
                         │                                 │
                         │              ack()              │
                         │<────────────────────────────────│
                         │                                 │
                         │         sendUserName()          │
                         │────────────────────────────────>│
```

alt

[valid condition]
    checkNameSuccess() (Server → Client)

[else]
    checkNameFailure() (Server → Client)

    newOrJoinGame() (Client → Server)

alt

[new game]
    sendGameCode() (Server → Client)

[join game]
    giveMeTheCode() (Server → Client)

    sendGameCode() (Client → Server)

    ack2() (Server → Client)

waitOtherPlayer() (Client → Server)

startGame() (Server → Client)
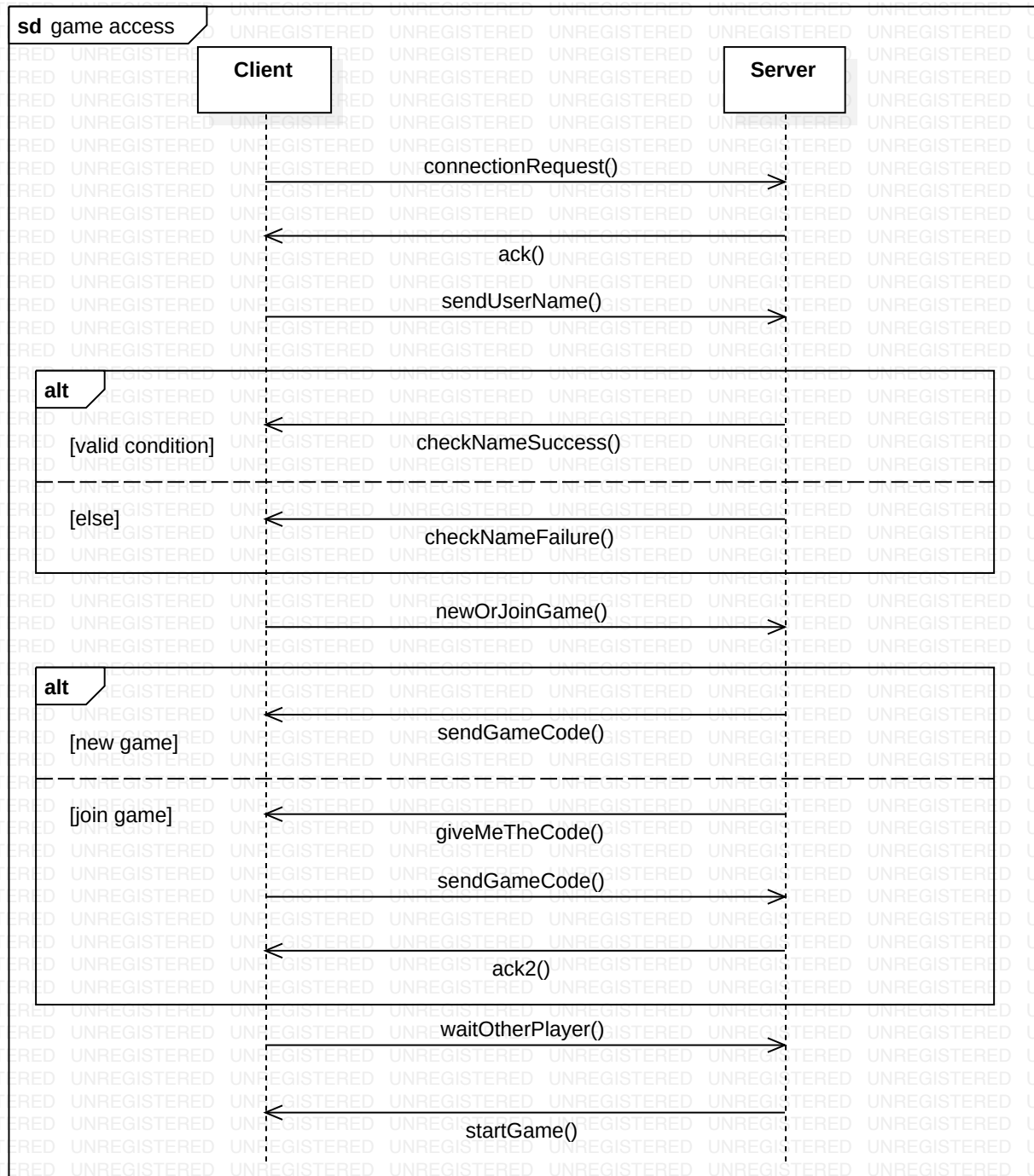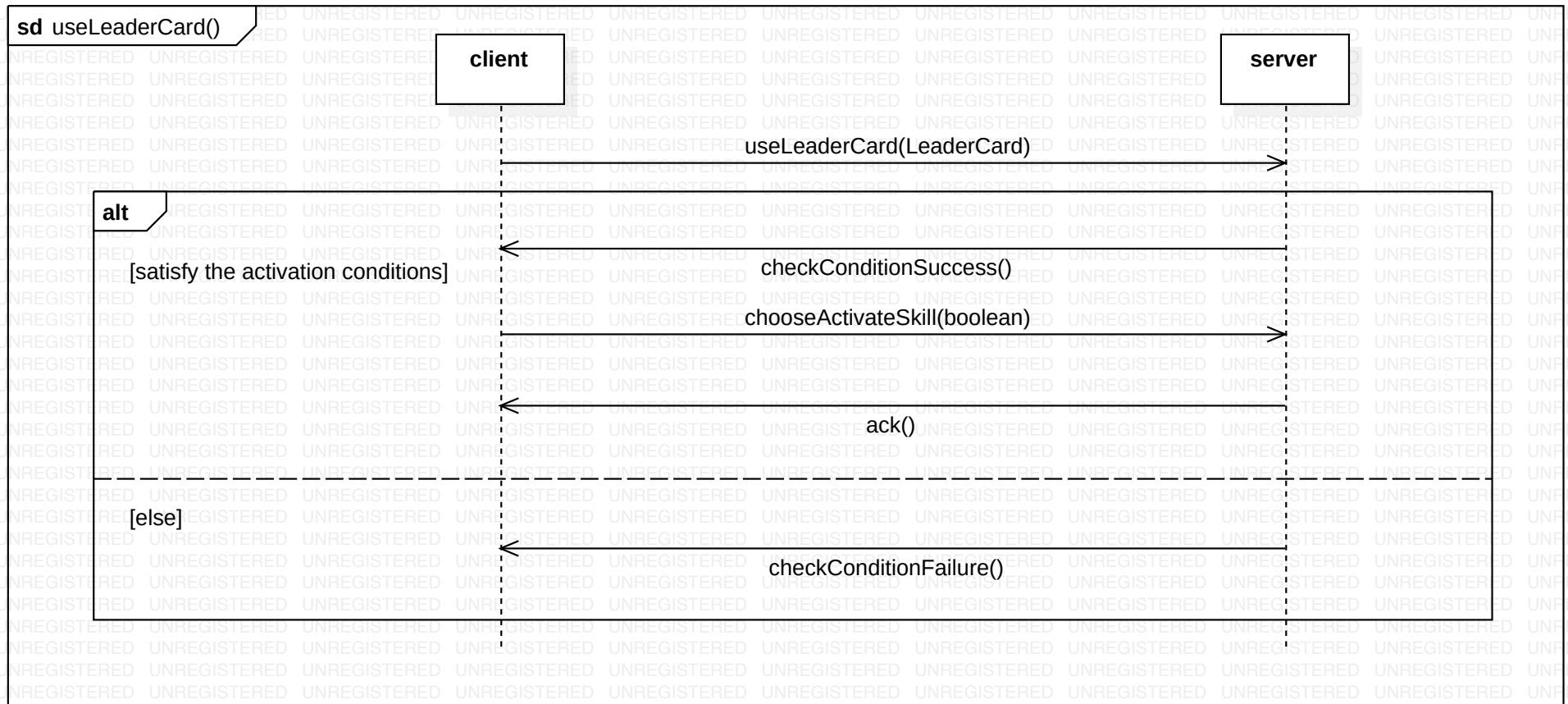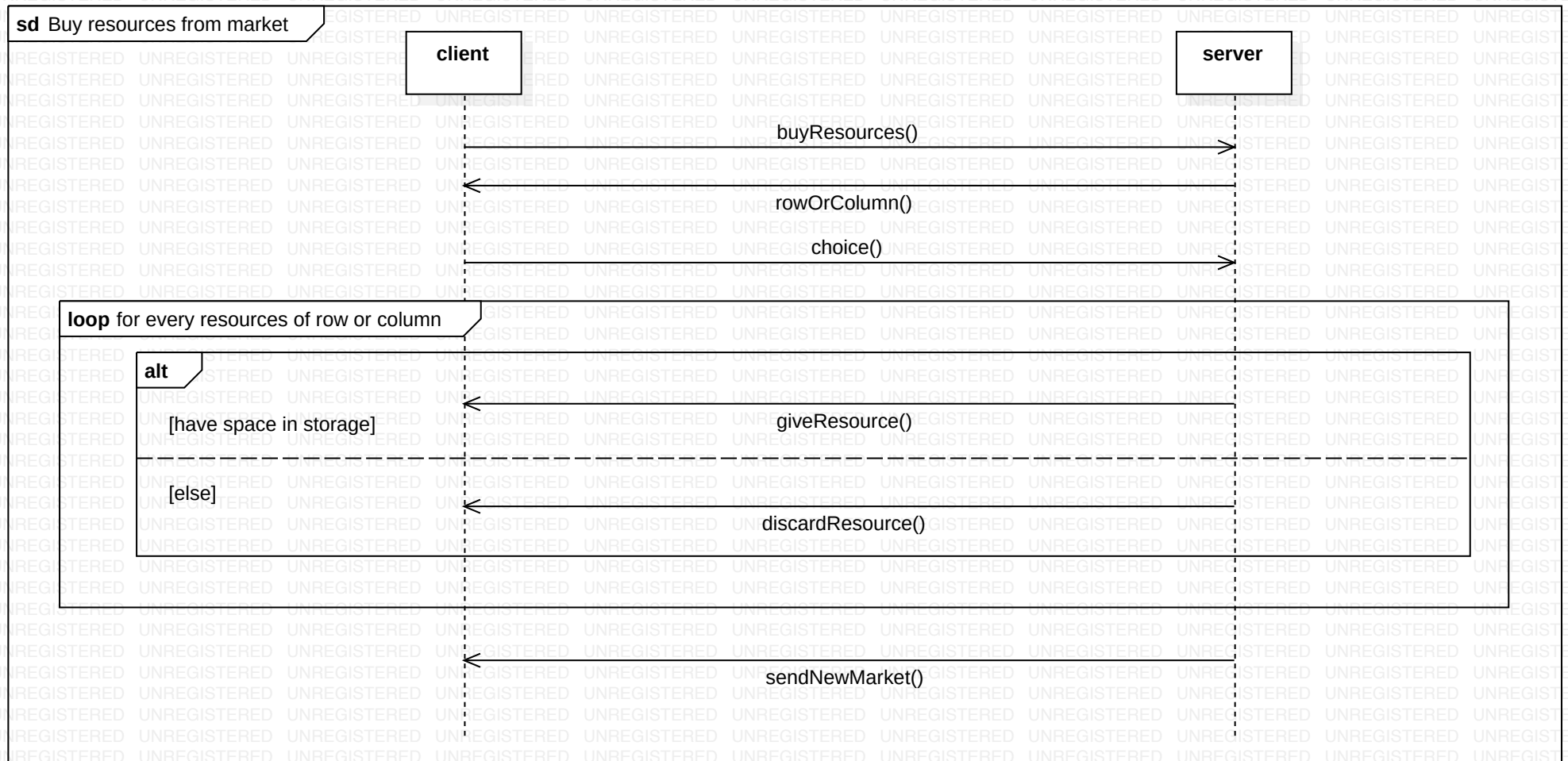

The client notifies its presence and requests the connection. The server responds with an ack.

The client enters the username and the server checks if the name is already in use in the game.

The Client chooses if he wants to start a new game or if he wants to join an already started one.

The player code is used to identify the game being played (in the case of multiple games at the same time).

In the case of multiplayer, the client waits for all players to log in and when all players are connected, the server notifies the start of the game

sd useLeaderCard()

client   server

useLeaderCard(LeaderCard)

alt

[satisfy the activation conditions]

checkConditionSuccess()

chooseActivateSkill(boolean)

ack()

[else]

checkConditionFailure()

The client chooses to use a leader card and notifies the server. The server checks that the player has the necessary requirements for activation. If so, it notifies the activation and the client can notify that it wants to activate the skill of the card. If not, the server notifies that the activation requirements are not satisfied.

sd Buy resources from market

client   server

client → server: buyResources()

server → client: rowOrColumn()

client → server: choice()

loop for every resources of row or column

    alt

        [have space in storage]   giveResource()

        [else]   discardResource()

server → client: sendNewMarket()

The client notifies that it wants to buy resources from the market and the server asks which row or column of the market it wants to buy.
The client makes its choice by notifying the server and the server check each resource and deliver it if there is space in the storage or discard it otherwise.
At the end of the loop the server send the new arrangement of the marbles in the market.