Eryantis Protocol Documentation

Elis Kina, Davide Osimo, Alessandro Fusè

Gruppo 38

Introduction

We designed our Client-Server communication protocol distributing the MVC Controller between both client and server interfaces.
Server can manage multiple requests from Client, identified by Player's nicknames. Server moves each management client connection to an instance of ClientHandler class.

Messages from Client to Server

CreateMatchRequest(nickname, numOfPlayers)

Client sends a CreateMatchRequest when he wants to create a new match.

Arguments

- nickname: client specifies his player nickname;
- numOfPlayers: client specifies the number of players of the match he wants to create.

Possible Responses

- CreateMatchResult(status, matchID, notes).

JoinMatchRequest(nickname, matchID)

Client sends a JoinMatchRequest when he wants to join an existing match.

Arguments

- nickname: client specifies his player nickname;
- matchID: client specifies the ID of the match he wants to join.

Possible Responses

- JoinMatchResult(status, notes).

AssistantCardRequest(assistantValue)

Client sends a AssistantCardRequest when he has decided the Assistant card to play.

Arguments

- assistantValue: is an integer from 1 to 10 that represents the value of the card chosen.

Possible Responses

- AssistantIsAvailable(result, notes);
- NextPlanPlayer(nickname, playerState.PLAN).

MoveStudentToDiningRoomRequest(student)

Client sends a MoveStudentToDiningRoomRequest specifying the student he wants to move into his dining room.

Arguments

- student: this is the student he has chosen.

Possible Responses

- MoveStudentResult(status, notes).

MoveStudentToIslandRequest(student, island)

Client sends a MoveStudentToIslandRequest specifying which student he wants to move into which island.

Arguments

- student: this is the student he has chosen;
- Island: this is the island he has chosen.

Possible Responses

- MoveStudentResult(status, notes).

MoveMotherNatureRequest(value)

Client sends this request when player is going to move Mother Nature

Arguments

- value: it stands for the number of islands that Mother Nature will run across.

Possible Responses

- MoveMotherNatureResult(status, notes).

CloudRequest(value)

Client sends this request when player is going to get students from the cloud.

Arguments

- value: it stands for the cloud's number chosen by the player.

Possible Responses

- CloudResult(status, notes).

Messages from Server to Client

CreateMatchResult(status, matchID, notes)

Server replies to client CreateMatchRequest with the result of his request.

Arguments

- status: specifies the result of the operation, is a Boolean;
- matchID: if status is set True, server specifies the ID of the match just created, else matchID is null;
- notes: if status is set False, server specifies one of the following issues:
    - NumOfPlayerNotValid: client chose to create a match with a number of participants different from 2 or 3.

JoinMatchResult(status, notes)

Server replies to client JoinMatchRequest with the result of his request.

Arguments

- status: specifies the result of the operation, is a Boolean;
- notes: if status is set False, server specifies one of the following issues:
    - NicknameNotValid: client chose a nickname already taken in the lobby he wants to join.

- MatchIDNotValid: client chose to join a match that doesn't exist or is started yet.

StartPlanification(nickname, playerState.PLAN)

Server sends StartPlanification to the client that is going to play his Planification Phase.

Arguments

- nickname: server specifies the player that is going to play first;
- playerState.PLAN: server sets the playerState of the first player.

AssistantIsAvailable(result, notes)

Client sends AssistantIsAvailable as a result of the card requested.

Arguments

- result: server specifies if the card selected is available, is a Boolean;
- notes: if result is false, server specifies one of the following issues:
  - CardAlreadyChosen: because another player chose that card before in that round;
  - CardNotFound: player chose a card that he has played yet.

StartAction(nickname, playerState.ACTION)

Server sends StartAction to the client that is going to play his Action Phase.

Arguments

- nickname: server specifies player that is going to play first in the Action Phase;
- playerState.ACTION: server sets to ACTION the state of the next player.

MoveStudentResult(status, notes)

Server sends the result of a player move.

Arguments

- status: is a Boolean, it's true if the move has been made
- notes: contains one of the following issues:
  - StudentNotFound: the client request specified a student that the player has not in his Entrance;
  - RowAlreadyFilled: player chose to move a student in his respective StudentRow which hasn't free spaces;
  - IslandNotFound: player chose an island that doesn't exist.

StartAction(nickname, playerState.ACTION)

Server sends StartAction to the client that is going to play his Action Phase.

Arguments

- nickname: server specifies player that is going to play first in the Action Phase;
- playerState.ACTION: server sets to ACTION the state of the next player.

CloudResult(status, notes)

Server sends CloudResult as a result of CloudRequest.

Arguments

- status: is a Boolean, it's False if the player chose a cloud that is not available.
- notes: server specifies CloudAlreadyChosen when another player got that cloud's students.

EndGame(nickname, notes)

Server sends EndGame to each client when is presented one of the following cases:

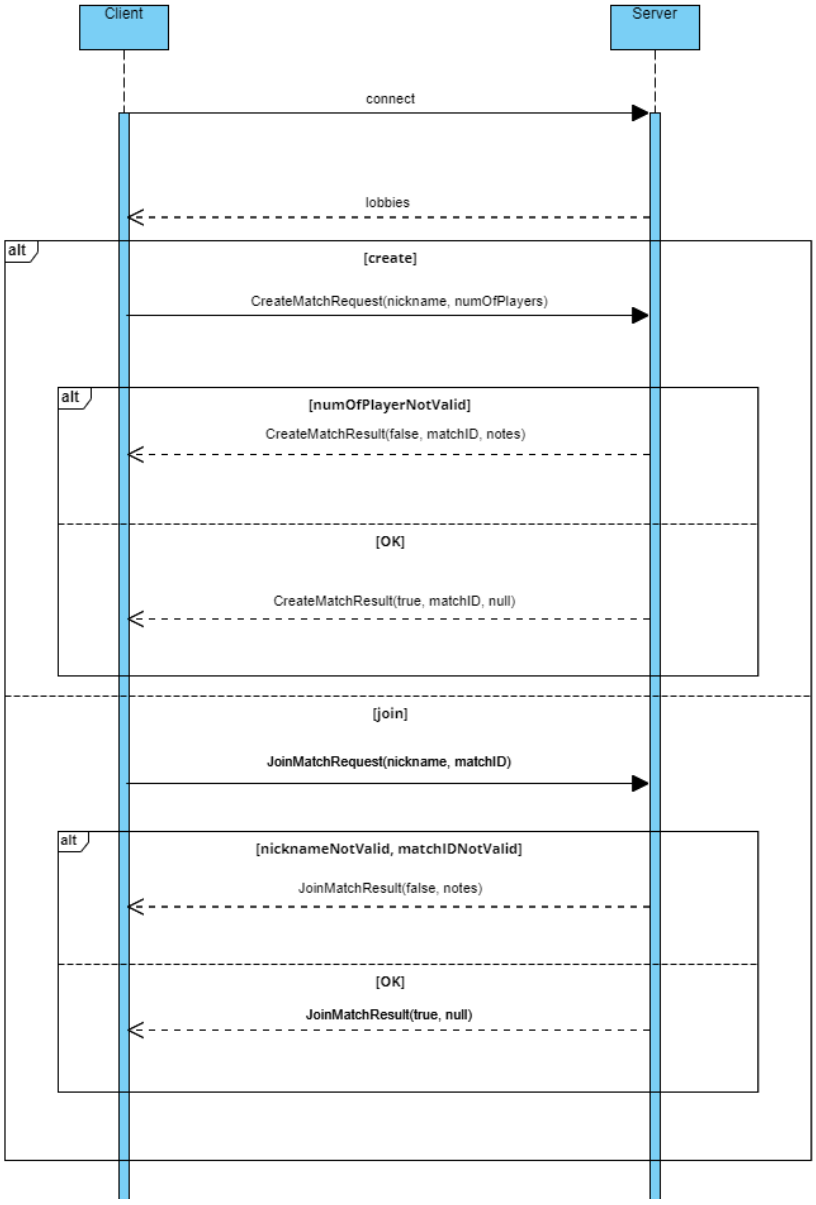
- a player builds his last tower;
- there are only three group of islands left;
- any player run out of his Assistant Card or bag is empty.

Arguments

- nickname: player's nickname;
- notes: server displays Winner message to the winner player(s).

Scenarios

Login phase



Planification Phase