Communication protocol

The communication protocol designed for the implementation of Eriantys board game uses Java ObjectStream for the exchange of information between the Server and the Clients.

It is designed in two layers. The lower one handles the ping/pong messages needed to check the connection between Client and Server and the higher one that exchanges the actual messages needed to play.

LOWER LEVEL

A ping message consists in a String “ping”, and a pong message in a String “pong”.

The server sends every 5 seconds a ping message and considers a client disconnected if it doesn’t receive any message in 15 seconds. When a Client receives a ping it responds with a pong.

HIGHER LEVEL

After the connection is established the client sends the nickname inserted by the user, the server checks if it’s available and responds accordingly. This is repeated until the nickname sent is valid. After this the server sends to the client the list of the new games waiting for other players, the list of previously saved games in reopening phase where the nickname of the client was one of the players and the list of saved games with the chosen nickname, if any is present. The client chooses a game from one of the lists or chooses to open a new game. The server responds confirming or denying the request (in case for example, during the choice other clients choose to enter the same game and the choice becomes not possible any more). The server keeps sending to the clients which have not already chosen a game all the available options every time they change.

From now on the client can send every game related message and the server will respond with a confirmation or with an error message that describes why the move is not valid.

In case of disconnection of a player (client) already in a game, all the other players in that game will receive a message informing them of the disconnection and the clients will close the sockets.

Every update to the board game made during a player’s turn will be sent to all the players. These updates include the students that are picked from the clouds, the students moved to the Chamber or to the Islands, the CharacterCard that is being activated.

All of the valid messages implement the Message interface, and in the next sections they will be listed and explained.

Messages from the client to the server

This messages include the set up messages and the actual messages needed to play.

SET UP MESSAGES

These are the messages to be sent to the server before a game begins:

Name of the class	Attributes	Description
NicknameMessage	String nickname	The message that contains the nickname chosen by the user
NewGameMessage	int numOfPlayers boolean expertMode	The message with the necessary data to create a new game

AddPlayerMessage	Int gameID	The message that contains the gameID of the game which the user wants to join
RestoreGameMessage	SavedGameData savedGameData	The message that contains the data of the saved game the user wants to restore

GAME MESSAGES

These messages implement the interface GameMessgae and they are the messages needed to make the moves accordingly to the game rules:

Name of the class	Attributes	Description
ChosenCardMessage	Card card	The message that contains the card chosen by the user
MoveStudentToChamberMessage	Clan clan	The message with the Clan of the student that the user wants to move from the hall to the chamber
MoveStudentToIslandMessage	Clan clan int islandIndex	The message that contains the Clan of the student that the user wants to move to an island and the index of that island
MoveMotherNatureMessage	int islandIndex	The message that contains index of the island where the user wants to move mother nature
ChosenCloudMessage	int cloudIndex	The message that contains index of the cloud chosen by the user
EndTurnMessage		The message after which the turn of the player is ended
ActivateCharacterCardMessage	CharacterCardID characterCardID	The message that contains the ID of the character that the user wants to activate
SetClanCharacterMessage	Clan clan	The message that contains the Clan needed for the effect of the activated character
ApplyCharacterCardEffectMessage1	int islandIndex	The message that contains the index of the island on which the user wants to activate the effect of the previously chosen character
ApplyCharacterCardEffectMessage2	int islandIndex Map<Clan, Integer> students1 Map<Clan, Integer> students2	The message that contains the necessary data for the effect of the chosen character card to be activated

Messages from the server to the client

These messages include the answers that the server sends to the clients after a move and the update messages needed to display the correct data on the view.

ANSWER MESSAGES

These messages include both the ones sent during the setup phase and the ones sent during the game:

Name of the class	Attributes	Description
NicknameAcceptedMessage	String nickname	The message with which the server informs the client that the chosen nickname has been accepted
AvailableGamesMessage	List<OpeningNewGameData> openingNewGameDataList List<OpeningRestoredGameData> openingRestoredGameDataList List<SavedGameData> savedGameData	The message with the data of the opening games, both new and restored, and the data of the saved games that the user can join
PlayerAddedToGameMessage	String message	The message with which the server informs the client that it has been added to the chosen game
GameStartedMessage	GameData gameData	The message that informs the client that the game chosen has started. It contains all the data needed to display the game on the view
GameOverMessage	List<String> winnersNickname	The messages that informs the clients that the game is over that contains le nicknames of the winners
GenericMessage	String message	The message that contains a generic string to display on the view
ErrorMessage	String error	The message sent to the client if a non valid move is performed or it sends to the server a message not valid for the game phase. It contains a String that describes the error
PlayerDisconnectedMessage	String DisconnectedPlayerNickname	The message that contains the nickname of a player that has disconnected

UPDATE MESSAGES

These messages are sent every time something changes in the model to keep the views of the clients up to date:

Name of the class	Attributes	Description
UpdateGamePhase	GamePhaseData gamePhaseData	The message that contains a GamePahseData object with all the information about the game phase and the character activated if any. This message is sent to all the players after every valid move

UpdateChosenCard	String playerNickname Card card	The message that contains the card chosen by the player with the nickname as attribute
UpdatePlayer	PlayerData modifiedPlayer	The message that contains the data of a player that has been modified
UpdateIsland	IslandData islandData	The message that contains the data of an island that has been modified
UpdataIslandManager	IslandManagerData islandManagerData	The message that contains the data of the island manager in case of a merge of islands
UpdateMotherNaturePosition	int islandIndex	The message that contains the index of the island where mother nature is
UpdateCloud	CloudData cloudData	The message that contains the data of a cloud that has been modified
UpdateCloudManager	CloudManager cloudManager	The message that contains the data of the cloud manager when the clouds are filled
UpdateCharacterCard	CharacterCardData characterCardData	The message that contains the data of a character card that has been modified