Personal Project – TypeRacing
Caden Kishimoto

TypeRacing JSON Protocol (For Requests/Responses)

Abbreviations: Leader Server = L, Game Server = G, Client = C

(Requests/Responses may be sent across different specific files, but the abbreviations define the groups that they belong to.)

Request: join (C -> L, G -> L, C -> G)

Initial join request for the sender

{

“type” : “join” – type is a join request

“clientPort” : “<int>” – OPTIONAL, only for when G is connecting to L (so L has reference)

“username” : “<String>” – OPTIONAL, only for C sending a name to G when connecting

}

Response: connected (L -> G, G -> C)

Leader responding/acknowledging to game server that connection is established

{

“type” : “connected” – type is a connected response

“message” : “<String>” – message saying connection established

}

Response: askForLogin (L -> C)

Leader responding to client to either sign-up or login (and acknowledging connection established)

{

“type” : “askForLogin” – type is an askForLogin response

“message” : “<String>” – contains menu to either sign-up or login

}

Personal Project – TypeRacing
Caden Kishimoto

Request: login (C -> L)

Client sends username and password as a sign-up/login

{

“type” : “login” – type is a login request

“username” : “<String>” – username of client

“password” : “<String>” – password of client

}

Response: showMenu (L -> C)

Leader responds to client showing the menu

{

“type” : “showMenu” – type is a showMenu response

“menu” : “<String>” – String of menu

}

Request: menuChoice (C -> L)

Client sends choice from menu options

{

“type” : “menuChoice” – type is menuChoice request

“choice” : “<int>” – choice of menu option

}

Personal Project – TypeRacing
Caden Kishimoto

Response: joinGame (L -> C)

Leader communicating to client that they're joining a game (connecting to a game server)

{

"type" : "joinGame" – type is joinGame response

"host" : "<String>" – host of the game server client is connecting to

"port" : "<int>" – port num of the game server client is connecting to

"message" : "Game found! Joining game…" – tell player a game is found and are joining

}

Request: askForMenu (C -> L)

Once a game is finished, client can request menu from leader again

{

"type" : "askForMenu" – type is askForMenu

}

Response: leaderboard (L -> C)

Leader showing leaderboard to client

{

"type" : "leaderboard" – type is leaderboard response

"leaderboard" : "<String>" – leaderboard as String

}

Personal Project – TypeRacing
Caden Kishimoto

Response: exit (L -> C)

Leader saying goodbye to client (acknowledging disconnect request by client)

{

“type” : “exit” – type is exit response

“message” : “Goodbye!” – saying goodbye to client

}

Request: answer (C -> G)

Client sending their answer (text) to the game server

{

“type” : “answer” – type is answer request

“answer” : “<String>” – client’s text answer

}

Response: answerCheck (G -> C)

Game server responds letting the client know it was either correct/incorrect

{

“type” : “answerCheck” – type is answerCheck response

“message” : “<String>” – finish info for correct, says try again for incorrect

}

Response: broadcast (G -> C)

Game server broadcasting to client a message (for either countdown info or a game start/over)

{

“type” : “broadcast” – type is broadcast response

“message” : “<String>” – message being broadcast to the client

}

Personal Project – TypeRacing
Caden Kishimoto

Request: disconnect (C -> G)

Client letting the game server know to disconnect/close connection resources

{

“type” : “disconnect” – type is a disconnect request

}

Request: checkGameState (L -> G)

Leader asking game server if its current game has finished

{

“type” : “checkGameState” – type is checkGameState request

}

Response: gameState (G -> L)

Game server telling leader its game state, which includes the number of players and if the current game has finished

{

“type” : “gameState” – type is gameState response

“finished” : “<boolean>” – true if game is finished, false if game is in-progress

“playerCount” : “<int>” – number of players in the current game

}

Request: gameWinner (G -> L)

Game server sending winner of a finished game to leader to update the leaderboard

{

“type” : “gameWinner” – type is gameWinner request

“winner” : “<String>” – username of the winner

}

Personal Project – TypeRacing
Caden Kishimoto

Response: error (L -> C, G -> C, L -> G)

Error responses for things like unknown requests, incorrect input, etc.

{

“type” : “error” – type is error response

“message” : “<String>” – error message depending on error type

}