Documento UML

Riva – Pieruz – Sartore – Tonoli

All'interno di questo documento viene spiegato il diagramma UML e i Sequence Diagram relativi alla parte di network del progetto.

Client:

Questa classe ha il solo scopo di chiedere all'utente se preferisce giocare tramite interfaccia testuale o grafica, creandone l'istanza.

PlayerBean:

Semplificazione della classe Player, presente nel model, che contiene tutte le informazioni necessarie per la corretta visualizzazione grafica del gioco.

Message:

Attributi:

Message_Type	Tipo del messaggio
Sender	Mittente

RMIClientHandler:

Contiene i metodi del client RMI che RMIServer ha la possibilità di chiamare.

RMIClient:

Attributi:

actionQueue	Coda di messaggi che permettono la desincronizzazione di RMI
processingAction	True se il client sta leggendo dalla coda
view	Riferimento alla GUI o TUI
server	Riferimento al VirtualServer ottenuto durante la connessione
commonResources commonGold starterCard	Copia dei rispettivi oggetti del model necessari per la resa grafica del gioco
Player	PlayerBean che rappresenta il giocatore
Opponents	Lista dei PlayerBean che rappresentano gli avversari

Metodi:

joinGame(int)	Permette di collegarsi al Game di indice {param}
setLobbySize()	Chiede al giocatore la dimensione della lobby che deve creare
flipCard(Card)	Richiede al server di girare la carta Card
placeStarterCard(Card)	Richiede di piazzare la starterCard
addToQueue(Message)	Aggiunge un messaggio alla coda
pickQueue()	Prende un messaggio dalla coda e chiama update(Message)
processQueue()	Se la coda non è vuota chiama pickQueue()
pingNetwork()	Risponde al ping proveniente dal server
update(Message)	Svolge le operazioni richieste dai vari messaggi

VirtualServer:

Contiene i metodi di RMIServer che il client può chiamare.

RMIServer:

La classe RMIServer si occupa di accettare le connessioni RMI, per poi istanziare una RMIConnection che sfrutterà per permettere la comunicazione tra client e controller.

Attributi:

actionQueue	Coda di azioni che il server deve svolgere
processingAction	True se il server sta processando la coda
server	Riferimento al Server principale
port	Porta su cui avviene la connessione
connection	Riferimento a Connection
controller	Riferimento a ServerController

Metodi:

login(RMIClientHandler, String)	Permette il login sul server principale
startServer()	Il server inizia ad accettare connessioni RMI
usernameTaken(String)	Ritorna true se l'username è già stato preso
flipCard(Card)	Chiama flipCard sulla Connection
placeStarterCard(Card)	Chiama placeStarterCard sulla Connection
setAchievement(Achievement)	Chiama setAchievement sulla Connection
addToQueue(Action)	Aggiunge una funzione lambda alla coda
pickQueue()	Prende una funzione lambda dalla coda
processQueue()	Se la coda non è vuota chiama pickQueue

Server:

La classe server rappresenta il server nel suo intero, crea due accettori di connessioni (RMIServer e SocketServer) per poter accettare entrambi i protocolli.

Attributi:

activeGames	Lista di Game incominciati e non finiti
startingGames	Lista di Game che stanno aspettando dei giocatori
client	Mappa che associa ad ogni Connection il suo username
controller	Riferimento al ServerController

Metodi:

startServer()	Crea un RMIServer e un SocketServer
login(Connection, String)	Aggiunge una Connection e il suo username alla mappa dei client
usernameTaken(String)	Restituisce true se l'username è già in uso
createLobby(String, int)	Crea una lobby di {param} giocatori e ci collega il player con {param} username

joinLobby(String, int)	Collega il player con {param} username alla lobby di indice {param}
getClientFromName(String)	Restituisce la Connection che ha il {param} username

ServerController:

Il ServerController permette al server di svolgere le azioni di createLobby e joinLobby.

SocketServer:

Creato dal metodo startServer della classe Server, il SocketServer accetta le connessioni tramite Socket dedicandogli un thread e un oggetto di tipo Connection, possiede infatti un solo metodo, ovvero startServer().

Connection:

Classe astratta che rappresenta la connessione tra Client e Server e permette la comunicazione tra questi ultimi grazie al protocollo scelto. Sul server svolge principalmente le azioni di un virtual client, dato che è la classe incaricata di chiamare i metodi del Controller per conto del Client. Questa classe viene ereditata da SocketConnection e RMIConnection, che ne forniscono una specializzazione.

Attributi:

isConnected	Booleano che rappresenta se il client è connesso o disconnesso

Metodi:

getConnectionStatus()	Ritorna lo stato della connessione
setConnectionStatus(Boolean)	Setta lo stato della connessione
setLobby(Controller)	Fornisce alla Connection il riferimento al Controller che deve chiamare
sendMessage(Message)	Invia un messaggio al client
joinGame(List<Controller>)	Data una lista di Game aperti ne fa selezionare uno dal client
createGame()	Permette al client di creare un nuovo Game
getUsername()	Restituisce l'username scelto dal Client
setAchievement(Achievement)	Setta l'obiettivo privato scelto dal Client

RMIConnection:

Estende la classe Connection per il protocollo RMI, vengono riportati solo le differenze rispetto alla classe padre.

Attributi:

client	Riferimento al Client
lobby	Riferimento al controller del Game a cui il client si è collegato
server	Riferimento al server principale
username	username del Client

Metodi:

update(Message)	Chiama funzioni del client in base al messaggio che gli viene notificato dal controller
ping()	Fa partire un ping per il client
pingClient()	Riceve la riposta al ping

SocketConnection:

Estende la classe Connection per il protocollo Socket, vengono riportati solo le differenze rispetto alla classe padre.

Attributi:

server	Riferimento al server principale
socket	Riferimento alla socket
in	Stream di input della socket
out	Stream di output della socket
lobby	Riferimento al controller del Game a cui il client si è collegato
username	Username scelto dall'utente

Metodi:

run()	Avvia il thread
onDisconnect()	Gestisce la disconnessione del client
onMessage(Message)	Svolge diverse funzioni in base al messaggio che riceve dal Client
update(Message)	Invia diversi messaggi al client in base al messaggio che gli viene notificato dal controller
ping()	Fa partire un ping per il client
pingClient()	Riceve la riposta al ping

SocketClient:

Attributi:

socket	Riferimento alla Socket
view	Riferimento alla view
in	Stream di input della socket
out	Stream di output della socket
disconnected	True se il client è disconnesso
commonResources commonGold starterCard	Copia dei rispettivi oggetti del model necessari per la resa grafica del gioco
Player	PlayerBean che rappresenta il giocatore
Opponents	Lista dei PlayerBean che rappresentano gli avversari

Metodi:

startClient()	Cerca di connettersi al SocketServer
readMessage()	Legge un messaggio in ingresso

sendMessage(Message)	Manda un messaggio al server
update(Message)	Aggiorna la view in base al messaggio ricevuto
onDisconnect()	Azioni per gestire la disconnessione del client