Peer-Review 2: Network

Kevin Ziroldi, Matteo Volpari, Samuele Villa, Tommaso Uberti Foppa

Gruppo GC07

Valutazione del protocollo di rete delle classi del gruppo 54.

Lati positivi

La descrizione, almeno testuale, del protocollo Socket, utilizzando dei messaggi (command pattern) è secondo una soluzione elegante.

Lati negativi

La descrizione fornita risulta essere molto poco dettagliata, inoltre dall'UML emerge come la connessione tramite socket sia ancora in stato embrionale.
Di seguito riportiamo le criticità che abbiamo potuto identificare sulla base di ciò che ci avete fornito.

Listener

Secondo il pattern MVC, i listener dovrebbero essere memorizzati nel model e l'aggiornamento dovrebbe partire dal model stesso ogni volta che questo si aggiorna.
Inoltre, la scelta di inviare i listener come parametro dei metodi del controller non è ottimale, dato che, questi dovrebbero essere memorizzati nel model e gli aggiornamenti devono essere inviati a tutti i listener e non in modo individuale. Anche se voi memorizzate i listener nel ListenerHandler, comunque non ha senso inviarli come parametro.

Metodi ClientRMI

In ClientRMI ci sono vari metodi, come createGame, joinGame, leaveGame e altri, che dovrebbero far parte del controller, il client dovrebbe solamente chiamarli, non possederli.

Messaggi client server

Nella descrizione, dite che ci sono una serie di messaggi che il client può inviare, ma non compaiono nell'UML come classi; inoltre avete scritto che il client RMI esegue chiamate di metodi sul controller.
Successivamente, nella descrizione dei Socket, dite che vorreste implementare questi messaggi con delle classi (command pattern).
A questo punto sarebbe opportuno sfruttare il command pattern anche in RMI, uniformando così i due protocolli.

Confronto tra le architetture

Abbiamo riscontrato alcune differenze nella gestione dei listener:

- Abbiamo previsto listener diversi per pezzi di modello diversi, al contrario di una sola interfaccia, che non permette specificità nell'invio degli aggiornamenti;
- Registriamo i listener nel modello al momento dell'ingresso della partita, senza inviarli ogni volta come parametro dei metodi;
- Abbiamo sfruttato il command pattern anche nella comunicazione tramite RMI, uniformando i due protocolli;
- Nella nostra implementazione, il client riceve una interfaccia VirtualServer e non il riferimento al controller, che nel nostro caso non è un oggetto remoto.