Peer-Review 2

Federico Rocca, Matteo Pasqual, Arianna Paone, Alberto Tonni

Gruppo 32

Valutazione del gruppo 22, riguardo a:

- UML delle classi
- Architettura di rete (Sequence Diagram)

Lati positivi

- L'implementazione del MainController come Singleton
- La corretta scelta di dare trasparenza alla rete tramite un'interfaccia poi implementata dai due protocolli di rete
- GameController e Observer sono generalmente corretti
- L'utilizzo delle interfacce per il calcolo dei punti e il controllo degli obiettivi sembra una soluzione promettente, poiché richiama lo Strategy Pattern consigliato in questi casi

Lati negativi

- Si suggerisce l'utilizzo di un'interfaccia al posto del ClientController, specificando i metodi che il client può chiamare e implementando tale interfaccia in RMIClient e SocketClient
- Sarebbe preferibile eliminare il passaggio attraverso MainController per le chiamate su GameController; per farlo, sarebbe necessario passare un riferimento di GameController al Client
- Gli Observer (Listener) dovrebbero essere chiamati esclusivamente dal Model (seguendo il Pattern MVC) e non dal Controller (seguendo il Pattern MVCC)

Confronto tra le architetture

La struttura della rete è in generale simile alla nostra.

Vi è una differenza nell'implementazione di Observer (per noi Listener): nel vostro progetto, le modifiche vengono comunicate al client attraverso un oggetto di tipo evento, mentre nel nostro caso si invoca un metodo sul client attraverso dei Notifier

L'impiego del tipo di classe Record per l'archiviazione dei mazzi, garantendo così la loro disponibilità costante, rappresenta un'interessante variazione rispetto alla nostra prassi, che prevede la ricreazione del mazzo ad ogni nuova partita.

Consigli

- Non è chiaro come il Client e il Server rilevino la disconnessione; si consiglia di implementare un segnale di ping.
- Per migliorare la leggibilità dell'UML, si consiglia di organizzare le classi in pacchetti e ordinare le frecce in modo più chiaro