Peer Review #2

Samuele Pischedda, Angelo Prete, Gabriele Raveggi, Andrea Sanvito
GC11

9 Maggio 2024

Revisione del MVC UML del gruppo GC01.

1 Aspetti Positivi

- Interfaccia comune tra socket e RMI.
- Corretto utilizzo dell'astrazione per la classe client.
- L'utilizzo di una classe astratta come azione (pattern command) è positivo per l'estendibilità.
- La divisione tra controller principale e controller della room permette di dividere le responsabilità e supportare più partite, poiché ogni controller può essere eseguito nel proprio thread.

2 Aspetti Negativi

- In generale, è meglio non inviare oggetti che son già contenuti nel `model` tra client e server, bensì un ID o una variabile che rappresenti quell'oggetto (e.g. le azioni `flipCard` e `playCard`, che hanno il parametro `PlayableCard`).
- Nelle azioni, il mainController potrebbe essere utilizzato come parametro del metodo `execute()` piuttosto che come attributo della classe azione. In questo modo, il comando può essere creato dal client piuttosto che dal server. Così, il client può inviare direttamente le azioni, anziché chiamare diversi metodi a seconda dell'azione che si desidera eseguire. In sintesi, il client avrà un metodo chiamato `sendAction(Action action)`, che invierà direttamente l'azione al server, invece di chiamare un metodo sul server per ogni specifica mossa che il client desidera effettuare.

3 Confronto tra le Architetture

- Noi abbiamo preferito utilizzare una connessione principale al `mainServer` per la gestione delle lobby, per avviare le partite e gestire con una nuova connessione per ogni partita in modo da dividere le responsabilità.
- Per la gestione delle notifiche inviate dal model, abbiamo creato una classe astratta `GameUpdate` (opportunamente implementata per i vari update) inviata al client tramite un unico metodo `send`.
- Per quanto riguarda l'implementazione dei socket, noi abbiamo deciso di emulare il comportamento di RMI tramite socket, consigliamo di usare l'interfaccia di Java `Serializable` in modo da poter inviare oggetti senza dover costruire da zero un parser.