Peer-Review 1: UML

Gabriel Voss, Nicola Tummolo, Francesco Zuliani, Andrea Tarabotto
Gruppo G36

10 maggio 2024

Valutazione del diagramma UML delle classi del gruppo GC26.

1 Lati positivi

Il gruppo GC26 ha adottato diverse scelte architetturali degne di nota:

- L'utilizzo di una PriorityQueue per le connessioni e di una queue nel GameController per gestire le modifiche al model, suggerisce un approccio efficiente nella gestione di operazioni concorrenti.
- L'adozione di un VirtualGameController, indipendentemente dalla tecnologia di rete utilizzata, evidenzia una buona modularità e flessibilità del sistema.
- La suddivisione dei compiti tra MainController e GameController favorisce la chiarezza e la separazione delle responsabilità.
- L'implementazione di thread separati per gestire le richieste asincrone è una soluzione efficace per garantire una maggiore reattività del sistema.
- L'utilizzo del pattern Command offre un'implementazione flessibile e manutenibile, che facilita modifiche future al codice.

2 Lati negativi

Tuttavia, ci sono alcuni aspetti che richiedono ulteriori considerazioni:

- L'uso di JSON potrebbe introdurre overhead di serializzazione e deserializzazione, soprattutto con grandi quantità di dati.
- È importante implementare strategie robuste per gestire situazioni anomale e garantire la stabilità del sistema.
- La fase di inizializzazione della connessione durante la creazione della partita potrebbe essere poco chiara, soprattutto riguardo alla scelta della tecnologia di rete.
- Potrebbe essere utile valutare l'implementazione di una coda anche sul client per migliorare la gestione delle richieste.

3 Commento ai flow diagram

Le caratteristiche del protocollo di rete descritto nella relazione rispecchiano la sequenza di chiamate nei protocolli di connessione:

MAIN SERVER

- L'implementazione sia di Socket che RMI è ben realizzata, garantendo una gestione efficace delle connessioni.

NOTIFY

- È apprezzabile l'utilizzo dell'interfaccia network come listener, dimostrando una buona progettazione nell'ascolto e nella gestione degli eventi di rete.

MAIN CONTROLLER

- Le richieste di connessione sono gestite in modo accurato, considerando tutti gli step necessari per l'inizializzazione del gioco attraverso Virtual RMI e Main Controller, garantendo un avvio fluido e senza intoppi.

MAIN CLIENT

- Il flusso decisionale tra l'utilizzo di Socket ed RMI risulta poco chiaro, richiedendo ulteriori dettagli o chiarimenti per una scelta consapevole e coerente.

4 Confronto tra le architetture

Nel confronto delle architetture tra il nostro approccio e quello del gruppo GC26, entrambe le proposte presentano similitudini e differenze significative.

- Entrambe implementano una coda per gestire le richieste e utilizzano interfacce per la comunicazione, garantendo modularità e flessibilità.
- La nostra scelta di implementare una coda sul client potrebbe offrire vantaggi aggiuntivi in termini di prestazioni.