Peer-Review 2: Sequence Diagram

<Kevin Wang>, <Pietro Sacchi>, <Maria Francesca Sfondrini>, <Valerio Cipolloni>

Gruppo <AM47>

Valutazione del diagramma UML delle classi del gruppo <AM04>.

Lati Positivi

- Utilizzo dell'interfaccia Serializable e della classe MessagePacket univoca e specializzata solo per la comunicazione via rete -> codice e logica più pulita;
- L'utilizzo di un Ack ad ogni inizializzazione o comunque un qualche tipo di riscontro a seguito di una azione via rete (anche se potrebbe essere non strettamente necessario per certe azioni come BoardInit o InitialHand).

Lati Negativi

- Poca scalabilità, seppur non implementiate la lobby come FA (il che porterebbe a gestire una moltitudine di connessioni), far gestire a ServerConnectionHandler sia l'accettazione delle connessioni in ingresso e per la spedizione e ricezione di messaggi durante la communicazione client-server non è ottimale, inoltre avere un altro thread solo per ServerSender sembra eccessivo (non basterebbe che sia direttamente il thread che gestisce la richiesta al server di occuparsi anche di spedirlo invece di delegarlo a qualche altro thread?);
- Problema di information hiding della interfaccia GeneralMessage... supponendo che la implementi il server le funzioni "ClientExecute" e "ServerExecute", e che sia incapsulata nella classe "MessagePacket" per poi spedirla via rete, si sta passando della logica e non solo i dati via rete;
- Mancato utilizzo dei costrutti speciali tipici per costruire un sequence diagram: opt, loop, case...;
- Mancanza di gestione dei casi "speciali"/imprevisti, come ad es: in caso non venisse rimandato indietro un ack in seguito al BoardInit, che cosa succede?

Confronto tra le Architetture

In generale la documentazione fornitoci della vostra soluzione sembra buona, eccetto la mancanza di certe raffinature in caso di imprevisti.

Similitudini:

- Utilizzo di una classe apposta specializzata solo per la comunicazione via rete.
- L'utilizzo di un "handler" generico, che il server può usare per communicare con il client in merito allo stato del gioco.

- La logica di comunicazione in linee generali tra server e client.

Differenze:

- Noi abbiamo più definito più la comunicazione basandoci sulla logica RMI (come, ad esempio, il problema della disconnessione che non manda una eccezione in questo caso (problema definito meglio su slack)), mentre voi con la comunicazione Socket, quindi sicuramente prenderemo spunto per certe logiche di comunicazione che avete implementato.
- Abbiamo definito più “corner case”.
- Noi utilizziamo Gson (libreria ben nota e molto utilizzata nell’ambito dei progetti) per la serializzazione e deserializzazione dei dati in Json.