Peer-Review 2: Network

Riccardo Polelli Filippo Pozzi Federico Quartieri Giacomo Tessera

Indice

Valutazione della parte di network del gruppo GC57 da parte del gruppo GC10.

Lati positivi

Durante la lettura e l'analisi dell'UML consegnato abbiamo riscontrato i seguenti lati positivi:

- Si apprezza particolarmente la chiarezza dell'implementazione delle classi nell'UML. Questo sicuramente sarà utile nella scrittura del codice per renderlo mantenibile e facilmente espandibile in caso di problemi e/o feature aggiuntive.
- La struttura scomposta del `controller`, con relativa interfaccia, permette di mantenere una struttura coerente e semplificata.
- Troviamo interessante la gestione degli errori con i codici, è molto professionale, espandibile e offre molta granuralità nella gestione degli errori.
- Apprezziamo molto l'organizzazione dei tipi di messaggio lato server (le 3 implementazioni) e lato client (le 5 implementazioni) e la scelta di serializzarli in un oggetto JSON per standardizzare la fase di serializzazione lato server e lato client.

Lati negativi

Abbiamo riscontrato anche alcune avvertenze:

- Non ci è chiaro come venga gestita l'inizializzazione del server, in quanto non sembra esserci un unico server, ma due oggetti distinti`rmiServer` e `socketServer`, e di conseguenza come vengano gestite contemporaneamente le richieste di client con implementazione diversa.
- Ci sembra scorretto (a meno di un obiettivo specifico) porre tutti gli attributi delle classi `Message` a public, dovrebbero essere private, ma potrebbe essere solo un problema di notazione.

Inoltre ci siamo accorti delle seguenti problematiche:

- `VirtualView` viene implementata solo da `SocketClientHandler` e `RmiClient` (che però hanno metodi in comune che non sono nell'interfaccia) e `SocketClient` non implementa `VirtualView`.
- `ServerSocket` e `ServerRmi` sembrano completamente diversi (`ServerRmi` non ha neanche controller e main).
- Ci sembra superfluo un `Message` di tipo `EndMessage`, in quanto un turno termina necessariamente con il pescaggio di una carta.

Confronto tra le architetture

- Interessante e molto *Java like* l'utilizzo di classi specifiche per ogni messaggio. Nella nostra implementazione ciò non avviene. In generale ci sembra un approccio valido e semplice da implementare ma al contempo molto specifico.
- Nella nostra implementazione non utilizziamo due server distinti, ma esiste un unico server che istanzia e pone in ascolo il `ServerRMI` e il `ServerSocket`.