# Peer-Review 2

Francesco Tranquillo, Denis Previtali, Andrea Zatti
Gruppo GC37

9 maggio 2022

Valutazione del diagramma UML del controller e protocollo di rete del gruppo GC47.

## 1 Lati positivi

1. Usare il pattern state per gestire il cambiamento di stato del client durante la partita è una buona idea.
2. Creare la classe lobby, separando questo compito dalla classe server, permette di non appesantirlo.
3. Possibilità di scegliere tra più lobby disponibili.

## 2 Lati negativi

1. UML rete:
   (a) Tutte le classi di tutti gli UML presentano solo attributi e metodi pubblici, non c'é nulla che limiti la visibilità di un qualsiasi parametro interno.
   (b) L'UML di rete risulta dispersivo e poco ordinato, sfavorendone la comprensione, che potrebbe essere ovviata dalla presenza di un testo descrittivo. Le classi non sono raggruppate per insiemi concettuali, ad esempio:

- nel pattern state gli stati sono posti anche molto distanti gli uni dagli altri;
- l'esistenza delle interfacce vuote "RequestMessage" e "ResponseMessage" usate probabilmente solo per dividere in due gruppi i messaggi quando dal punto di vista del codice basterbbe usare due packages diversi mentre nell'UML raggruppare fisicamente in due zone distinte le classi dei messaggi; se invece fossero le interfacce standard delle librerie di java sarebbe necessaria l'implementazione di tutti i metodi di queste ultime, di cui numerosi metodi sarebbero inutili;
- per quanto riguarda il pattern observer non è chiaro chi sia l'osservabile e chi l'osservatore, che potrebbe essere risolto disponendo meglio le classi interessate da questo pattern.

(c) Mettere sul controller un metodo distinto per ogni carta personaggio è contro l'orientamento agli oggetti di java e ne comporta la perdita di scalabilità, tanto è vero che nel model non sono presenti classi carte personaggio, eccezion fatta per quelle che possiedono una lista di studenti.
Anche un classe messaggio per ogni carta è ridondante in quanto è possibile raggrupparne alcune avendo gli stessi input.

(d) Non è chiaro chi orchestra il flusso sequenziale del gioco, oppure chi triggera gli eventi in caso di un'azione (L'unica spiegazione che abbiamo ipotizzato è che il metodo recive venga lanciato dall'esterno su un thread dedicato per poi non terminare mai raccogliendo tutti i messaggi in ingresso).

(e) L'esistenza sia di un pattern state che di un enum per lo stato del client è ridondante, sarebbe meglio conservarne solo uno dei due.

2. Sequence diagram:

(a) Non è chiaro se i messaggi scambiati siano i messaggi di rete oppure chiamate di funzioni, in questo ultimo caso il messaggio dovrebbe riportare il nome della funzione a cui si riferisce.

(b) Nel sequence diagram non è presnete un messaggio di errore nel caso in cui un client tenti di connettersi ad una lobby che nel frattempo si è già riempita, questo è dovuto anche al fatto che

la lista di lobby che viene mostrata al client è potenzialmente un dato obsoleto essendo un duplicato di quella nel server.

# 3 Confronto tra le architetture

L'esistenza di una lista di lobby e della possibilità di scegliere a quale accedere è una buona funzionalità che potremmo adottare.