Peer-Review 1: UML

Carlo dell'Orco, Federico Falcone, Francesco Cugliandro, Jacopo Donnini

Gruppo 12

Valutazione del diagramma UML delle classi del gruppo 21.

Lati positivi

Dopo aver esaminato il codice del progetto, abbiamo notato che sono presenti alcune scelte di design che dimostrano una buona conoscenza dei principi di programmazione orientata agli oggetti. In particolare, riteniamo intelligente l'utilizzo della classe PlayerHand per gestire l'ordinamento delle tessere da inserire nella shelf. Inizialmente, avevamo pensato di gestire questa operazione all'interno del controller, ma la soluzione adottata dai programmatori è decisamente più elegante e funzionale.

Anche la classe PlayerManager, all'interno del match, è stata implementata con efficacia: questa classe consente di gestire in modo efficace i giocatori in attesa dell'inizio della partita, dovendo il primo giocatore specificare la dimensione della lobby e attendere il collegamento di tutti i giocatori necessari. Questo approccio garantisce un alto livello di controllo sulla partita e consente di prevenire eventuali problemi di connessione o di compatibilità tra i giocatori.

Per quanto riguarda le due classi controller (PlayerController e GameController), sono state gestite in modo accurato, con la presenza di molti metodi effettivamente utili e implementati anche da noi. Inoltre, la scelta di includere interfacce ben strutturate e razionali, che presentano tutti i possibili metodi che verrebbero invocati direttamente da un utente. Tuttavia, abbiamo notato che alcune opzioni potrebbero essere superflue o addirittura indesiderate, come ad esempio la funzione pauseGame(), che non dovrebbe essere possibile attivare in determinati contesti.

Infine, abbiamo notato che l'idea di gestire gli input del client con un handler separato dalla chat potrebbe essere utile, ma ci sono alcuni aspetti che richiederebbero un'ulteriore riflessione e un maggiore dettaglio nella documentazione. In particolare, mancano i metodi del server per lo scambio di messaggi, il che potrebbe creare alcune difficoltà di implementazione.

Lati negativi

Riteniamo che la scelta di tipologia di dati non sia ottimale: sebbene la parte relativa a controller, handler e game sia abbastanza chiara, la scelta di rappresentare i dati ci è sembrata fin troppo dettagliata e complessa per il modello, con alcune classi o interfacce che sembrano superflue.

In particolare, l'idea di avere una sovraclasse astratta Card per gli obiettivi privati e comuni non è del tutto da escludere, anche noi avevamo inizialmente pensato di gestirli in questo modo. Tuttavia, la scelta di avere anche le ItemTileCard e le ScoringTokenCard come sottoclassi di questa sovraclasse non è conveniente né dal punto di vista del modello, né dall'astrazione dei dati. Inoltre, l'enumerazione ScoringToken è abbastanza inutile, in quanto rappresenta semplicemente degli int.

Un altro aspetto che non ci convince è la scelta di avere le Cell all'interno della shelf e della board. A nostro parere, sarebbe sufficiente avere direttamente una matrice di ItemTileCard, dato che anche la classe Cell non è altro che una matrice.

Purtroppo, non abbiamo avuto una descrizione dettagliata del funzionamento del modello, motivo per cui le nostre critiche potrebbero essere frutto di incomprensioni. Tuttavia, sulla base della documentazione disponibile, abbiamo ritenuto opportuno esprimere i nostri dubbi e le nostre perplessità riguardo al modello proposto

Confronto tra le architetture

Nel complesso, siamo d'accordo con la maggior parte delle scelte fatte per il modello di dati proposto, ad eccezione dei punti deboli che abbiamo già evidenziato in precedenza. Nel nostro flusso di ragionamento, abbiamo seguito un approccio simile per la maggior parte delle funzionalità. Tuttavia, poiché abbiamo deciso di non implementare funzionalità multipartita, ci sono alcune differenze nella gestione dei Match (che abbiamo chiamato "Game").

Inoltre, ci sono alcune enumeration che abbiamo deciso di non includere, poiché le abbiamo ritenute evitabili. Tuttavia, riteniamo che la rappresentazione degli stati attuali della partita attraverso le enumeration Turn e Game Phases sia un ottimo modo per gestire lo stato attuale della partita.

Rispetto al nostro controller, quello proposto è molto più dettagliato. Inizialmente avevamo dei dubbi sulla sua implementazione, ma la soluzione proposta nel modello ci sembra funzionale ed efficiente. Anche il ClientInputHandler è molto ben descritto e concettualizzato nel modello proposto, cosa che invece abbiamo solo vagamente immaginato finora. Tuttavia, il Server non ha ancora tutti i metodi descritti, ma riteniamo che la sua implementazione sia essenziale per il funzionamento del gioco