Peer-Review 1: UML

Mattia Colombo, Piervito Creanza, Simone Curci, Marco Febbo
IS24-AM02

29 marzo 2024

Valutazione del diagramma UML delle classi del gruppo IS24-AM11.

1 Lati positivi

- **Strutture dati adeguate:**
  `HashMap<Position, CardContainer>` per tracciare le posizioni delle carte nel campo di gioco e la capacità di pescare le carte dal terreno o dai mazzi con `addCardToHand` dimostrano un'attenta progettazione delle strutture dati.

- **Gestione inzializzazione partita e giocatori:**
  L'UML illustra efficacemente il processo di inizializzazione, compresa la scelta della side della `StarterCard` e la pesca automatica delle carte risorsa e oro, suggerendo un flusso di gioco chiaro fin dall'inizio.

- **Calcolo punti carte obiettivo:** Il metodo `countObjectivesPoints` fornisce un meccanismo solido per il calcolo dei punti derivanti dalle `ObjectiveCard`, un aspetto chiave nella determinazione del punteggio nel gioco.

- **Classe PlayerBoard ben definita:** La rappresentazione dettagliata dell'istanziazione del giocatore e della `PlayerBoard` indica un'attenta gestione dei dati del giocatore e del suo stato nel gioco.

2 Lati negativi

- **Overengineering di BasicRuleSet e RuleSet:** L'introduzione di `BasicRuleSet` e `RuleSet`, benché nobile nell'intento, rappresenta un overhead non necessario e non corrisponde a nessuna de;;e feature aggiuntive richieste.

- **Overengineering in uso di Guava?**
  L'impiego della libreria Guava per rendere immutabili le variabili delle carte appare come un overengineering, specialmente se non esistono metodi per modificare gli attributi delle carte.

- **Assenza di meccanismo per mischiare i mazzi:** Non sembra esserci un modo chiaro per fare shuffling dei mazzi di carte.

- **Complessità nella Modelizzazione delle ObjectiveCard:**
  Le `ObjectiveCard` appaiono eccessivamente complicate, i collegamenti UML poco chiari.

- **Calcolo punteggio carte oro:**
  Nonostante esista `countObjectivesPoints`, manca un equivalente per contare i punti delle carte oro, limitando la valutazione completa dei punti.

- **Ambiguità nella rotazione delle Carte:**
  La presenza dell'`isRotatedFlag` non chiarisce come un giocatore possa effettivamente ruotare una carta nella sua mano prima di piazzarla.

- **Nomi non chiari, concetti ambigui:**
  Concetti come `getPlateau` e la gestione delle risorse sul campo del giocatore sono poco chiari. Non è evidente cosa rappresenti esattamente il 'plateau': se sia inteso come il livello più alto di carte già piazzate, le carte esposte, o se abbia una funzione specifica come identificare i corner liberi per il piazzamento di nuove carte. Questa ambiguità può portare a confusione nella comprensione e nell'implementazione delle meccaniche di gioco.

- **UML troppo esteso, interfacce poco utili:**
  L'UML appare eccessivamente esteso con l'utilizzo di interfacce che non sembrano portare un valore aggiunto significativo. Consigliamo di

non implementare interfacce in relazione 1:1 con le classi, ma piuttosto implementare i metodo richiesti direttamente nella classe stessa. Si consiglia di adottare il **principio KISS (Keep It Simple, Stupid)** per semplificare il diagramma e renderlo più comprensibile.

- **Mancanza controllo punti per modalità Armageddon:**
  Non è chiaro chi controlli se un giocatore ha superato 20 punti per attivare la modalità `toArmageddon()`.

- **Relazione tra GameLogic e Ruleset**
  Nonostante le classi GameLogic e Ruleset risultino in relazione, non è chiaro in che modo quest'ultima sia realizzata. Forse la classe BasicRuleset dovrebbe essere in composizione dentro GameLogic.

3 Confronto tra le architetture

Non sono stati riscontrati scelte architetturali o design patterns con maggiore forza rispetto al nostro progetto. Tuttavia, abbiamo notato nella nostra implementazione l'assenza di un attributo che permetta di salvare il colore della pedina del giocatore, come invece presente nel diagramma UML del gruppo IS24-AM11.