

# Peer-Review 1: UML

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

2 aprile 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 inizializzazione 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'istanziamento del giocatore e della `PlayerBoard` indica un'attenta gestione dei dati del giocatore e del suo stato nel gioco.

- **Ottima gestione dei Deck** attraverso un'interfaccia unica e l'utilizzo del **design pattern** `Factory` per la creazione degli stessi.

## 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 delle 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.
- **Complessità nella Modellizzazione delle ObjectiveCard:**  
Le `ObjectiveCard` appaiono eccessivamente complicate, i collegamenti UML poco chiari.
- **Calcolo punteggio carte oro:**  
Nonostante esista l'enum `PointsRequirementsType` che permette di differenziare fra i tipi di carte Oro, non è chiara dal diagramma l'esistenza di metodi che contino gli angoli coperti dalla carta piazzata. Questa assenza potrebbe rendere difficile il calcolo dei punti dati al giocatore dal piazzamento di una carta oro.  
Esiste `Players.CardContainer coveredCorners`, il quale essendo però collegato a `PlayerField` fa pensare più a un concetto globale, piuttosto che proprio di ogni carta, il che sarebbe necessario per poter correttamente contare i punti.
- **Ambiguità nella rotazione delle Carte:**  
Non è chiaro come si possano piazzare le `PlayableCard` capovolte e come venga gestita la rotazione delle stesse.
- **Eccessivo numero di classi:**  
Esistono molte classi che potrebbero essere eliminate o unite in una sola, come ad esempio `Plateau` e `PickableTable`. La ridondanza di classi può portare a un aumento della complessità e della manutenibilità del codice.
- **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 metodi richiesti direttamente nella classe stessa. Si consiglia di adottare il **principio KISS (Keep It Simple, Stupid)** per semplificare il diagramma e renderlo più comprensibile.

- **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 architetture 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.