Peer-Review 1: UML

Sciaraffa, Setaro, Sposito, Testa

Gruppo GC41

Valutazione del diagramma UML delle classi del gruppo GC31.

Lati positivi

In generale l'UML si presenta molto ordinato ed è abbastanza completo nella rappresentazione del gioco.

Package Player

- Buona l'idea di utilizzare una mappa per immagazzinare l'informazione relativa alle posizioni delle carte giocate.

Package JsonAdapter

- Abbiamo apprezzato l'utilizzo del design pattern Adapter per il Parser JSON.

Package Exceptions

- Ottima la scelta di implementare delle exceptions specifiche per il proprio progetto, in modo da garantire una migliore chiarezza nella gestione delle stesse.

Package Strategy

- Condividiamo la scelta di aver utilizzato uno strategy pattern per il calcolo dei punti assegnati ad ogni singolo obiettivo completato durante lo svolgimento della partita.

Package Card

- Ci appare molto utile l'idea di utilizzare classi differenti per il front e il back di una carta.

Lati negativi

La descrizione testuale risulta essere un po' carente, pertanto alcune scelte non ci risultano del tutto chiare.

Package Player

- Non ci appaiono evidenti i benefici del restituire un oggetto di tipo 'GameModel' nei metodi del GameModel;
- Manca un 'gameState' che differenzi le varie fasi della partita;
- Non appare evidente l'utilità dell'attributo 'pawnSelector' nel GameModel.

Package Controller

- Non ci è chiaro come avvenga la ricerca di un GameController all'interno della 'gamesList' del Controller generale. Una possibile miglioria potrebbe essere quella di sostituire la 'gamesList' con una Map<ID, GameController>.

Classe Server

- Poco autoesplicativa. Manca la descrizione testuale della classe.

Classe Deck

- Non ci è chiaro il motivo per cui le informazioni relative al path dei file JSON siano immagazzinate come attributi nella classe Deck, dal momento che per costruire le carte occorre quell'informazione in primo luogo.

Package Enumeration

- Non siamo certi della scelta di "mischiare" i colori delle pedine e quelli delle carte. Inoltre, il colore "Black" dev'essere, secondo noi, considerato come attributo addizionale del giocatore, non come colore della pedina.

Package Card

- Non capiamo la scelta di far ereditare alla classe Objective Card la classe astratta Card, in quanto crediamo che i suoi attributi non siano necessari al calcolo dei punti.

- Seguendo la stessa linea di pensiero, se la classe Objective Card fosse resa una classe a sè stante, si potrebbe far collassare verso l'alto le classi Playable Card e Card.

Package Objective

- Nella sottoclasse Count abbiamo dei dubbi sull'attributo resource: se l'obiettivo fosse pubblico non ci sarebbe modo di tenere traccia delle risorse per ogni giocatore. E anche se l'obiettivo non fosse pubblico, ma relativo ad un singolo giocatore immaginando che l'obiettivo venga istanziato all'inizio della partita, non ci è chiaro in che modo la classe Count possa capire il numero di risorse necessarie per ogni obiettivo. La classe Count risulta poco esplicativa e difficile da capire.

- Nella sottoclasse CoverCornerScore è necessario ricordare la posizione in cui la carta è stata giocata per poter calcolare il relativo punteggio dato dalla copertura degli angoli delle carte adiacenti.

Confronto tra le architetture

Condividiamo con il vostro gruppo una buona parte delle scelte progettuali, come ad esempio quelle in merito alla gestione del tabellone e l'utilizzo del pattern strategy per le carte obiettivo. A

differenza vostra, la nostra architettura presenta enumerazioni più specifiche (e. g. distinzione tra PlayerColor e CardColor), atte a garantire una maggiore chiarezza. La gerarchia dell'entità carta viene gestita dai 2 gruppi in maniera differente. A nostro avviso, come fatto notare sopra, sarebbe meglio separare le carte objective dal resto delle carte. La gestione della logica di gioco è invece abbastanza simile nelle 2 architetture. In conclusione, evidenziamo ulteriori punti di forza da noi apprezzati e non presenti nella nostra architettura, ma che riconosciamo essere una possibile migliora del nostro progetto (i. e. eccezioni personalizzate, utilizzo del pattern Adapter).