Peer review 1: class diagram

Simone Corbo, Valeria De Gennaro,
Matteo Delton, Beatrice Di Raimondo Metallo
Gruppo AM13

3 aprile 2023

Valutazione del diagramma UML delle classi del gruppo AM22.

1 Lati positivi

Il controller risulta ben ingegnerizzato e scandisce chiaramente le varie fasi di gioco: è presente una classe che si occupa della creazione del gioco (`GameCreation`), una che gestisce i turni in generale (`Turn`) e una che gestisce le operazioni che possono essere effettuate solo durante il proprio turno (`OnlyDuringTurnMoves`). La partita nel complesso è gestita da `Game`, che fa da tramite tra controller e model.

Apprezziamo inoltre l'utilizzo della classe `PointsControl` che raggruppa tutti i metodi finalizzati al calcolo del punteggio, e quello della classe `Points`, che permette di tenere traccia del punteggio acquisito dal giocatore e di sapere come sono stati ottenuti i punti (se con una `PersonalGoalCard`, con una `CommonGoalCard` o con le adiacenze).

La classe `Player` poi ci sembra ben implementata: ha tutti gli attributi necessari alla definizione del giocatore (ad esempio, ad ogni giocatore corrisponde una `Bookshelf` e ogni giocatore ha un booleano che specifica se è il primo a giocare).

2 Lati negativi

Per quanto concerne le carte obiettivo personale, riteniamo che l'approccio utilizzato sia eccessivamente dispendioso, in quanto si potrebbe fare uso di

una struttura dati finalizzata alla rappresentazione della singola carta, da inserire successivamente all'interno di un mazzo. Così facendo, si potrebbe pescare la carta e usare un metodo finalizzato a calcolare il punteggio acquisito dal giocatore.

Per quanto non sbagliata, la creazione di 12 classi distinte per ogni possibile configurazione degli obiettivi comuni è evitabile: è possibile parametrizzare le classi in modo da ridurne il numero e semplificare i metodi di controllo.

La classe `EndGameControl` potrebbe essere eliminata, sostituendola con due booleani all'interno delle classi che ne fanno uso, come anche la classe `Bag`, che non sembra offrire vantaggi rispetto alla sola presenza del suo attributo `content` (implementato come `ArrayList<TilesType>`).

3 Confronto tra le architetture

Abbiamo riscontrato diverse soluzioni in comune, come la presenza di una classe da cui ereditano i singoli obiettivi comuni e l'utilizzo di un'enumerazione per definire i vari tipi delle tessere. A questo proposito ci sembra valida l'idea di utilizzare un elemento in più, `EMPTY`, per identificare le celle vuote; noi invece facciamo uso degli `Optional`, che però richiedono una gestione più laboriosa che ci saremmo potuti risparmiare con la soluzione adottata dal gruppo AM22.

Nel nostro UML ci siamo concentrati maggiormente nell'organizzazione del model, cercando in particolare di generalizzare gli obiettivi comuni, tralasciando una migliore organizzazione del Controller, come invece è stato fatto dal gruppo AM22, che l'ha saputo separare chiaramente dal model e ha pensato alle varie situazioni in cui dovrà intervenire.

Per migliorare la nostra architettura cercheremo di individuare con chiarezza i momenti di gioco per poi traslarli nel controller.