Peer-Review 1: UML

A. Pellicari, P. Ravasi, F. Tagliabue, G. Versiglioni
Gruppo 50

3 aprile 2024

Valutazione del diagramma UML delle classi del gruppo 40.

1 Lati positivi

1.1 PointsMaker

- Sistema unico per la gestione di punteggi e obiettivi (meno duplicazione del codice);
- Utilizzo di uno stesso algoritmo e parametrizzazione per risolvere casi di punteggi differenti (e.g. SymbolsCountAdder, DiagonalLayoutAdder, TwoColorsLayoutAdder), invece di utilizzare una enumerazione contenente tutti i possibili casi

1.2 Card

- Uso di metodi funzionali che rendono il codice più robusto (e.g. getCorners restituisce una copia degli angoli creando un nuovo array)

1.3 Game

- Tutti i metodi dichiarati nella classe Game garantiscono l'impostazione delle carte e del tavolo a inizio partita, e (in generale) un corretto flusso di gioco

1.4 Player

- Distinzione tra punti ottenuti nel corso della partita e punti ottenuti grazie agli obiettivi (utile per la risoluzione di pareggi)

2 Lati negativi

2.1 PointsMaker

- Impossibilità di verificare il numero di punti che si otterrebbero prima del piazzamento di una carta

2.2 Card

- Metodo canBePlaced in ResourceCard e StarterCard non avrebbe alcun senso di esistere se non fosse per la presenza di GoldCard
- Poca flessibilità a fronte dell'introduzione di nuove carte senza cambiare le regole (e.g., carte con risorse centrali anche sul fronte, carte iniziali aventi un colore, carte con punteggi anche sul retro...)
- In CardType l'enumerazione è troppo specifica con Revealed_Resource_0,1 e Revealed_Gold_0,1 (troppo dipendente dal gioco implementato)

2.3 Game

- Si può usare la classe DeckManager per gestire sia i deck di gioco che le carte rilevate (altrimenti, ripetizione)

2.4 Player

- Impossibilità di avere contemporaneamente un token colorato e quello nero
- Poiché nell'enum Symbol sono definiti anche i valori Empty e None, presenza di mappature inutili e insensate in numSymbols

2.5 Chat

- Classe Message non modella la possibilità di un messaggio di essere di tipo broadcast (cioè indirizzato a tutti) oppure di essere mandato ad un giocatore specifico

3 Confronto tra le architetture

- Il nostro model ha una matrice di dimensione fissa per memorizzare la posizione degli angoli, mentre in questo model si usa una struttura dinamica che memorizza la posizione degli angoli come la somma della posizione della carta (assoluta) e della posizione dell'angolo nella carta (locale)

- Noi non abbiamo implementato una classe DeckManager per la gestione dei mazzi, ma li abbiamo gestiti (creazione mazzi e metodi relativi) nella classe Game

- Nella prima versione dell'UML non avevamo una classe per gestire istanze di gioco come la loro classe Menu, quindi abbiamo implementato un singleton che si occupa di creare istanze di Game e memorizzarle in una lista