Peer-Review 1: UML

Alessio Spineto, Nicoló Solbiati, Alessandra Romano, Clara Spezzi
G39

30 marzo 2023

Valutazione del diagramma UML delle classi del gruppo G29.

1 Lati positivi

Nel complesso la vostra architettura è molto semplice ma funzionale.

1.1 Classe Board

Abbiamo trovato molto interessanti i metodi refill() e refillNeeded() separati, in modo da poter invocare il primo solo se il secondo dà come risultato true. Inoltre riteniamo sia stata un'ottima scelta l'aver implementato il metodo takeableTiles(), che noi tuttavia pensiamo di implementare nella View.

1.2 Classe Bookshelf

Ottimo il metodo isFull() (che noi ci siamo dimenticati di implementare e che implementeremo sicuramente!) ma, soprattutto, il metodo calculateScoreAdjacentItemTiles().

1.3 Controller

Riteniamo che il metodo addNewPlayer(String s, int i) sia utile e necessario da inserire nel Controller.

2 Lati negativi

Mancano i nomi delle relazioni e le relative cardinalità.

2.1 Classe CommonGoalCard e inerenti

Riteniamo che sia inutile avere una classe CommonGoalCard che implementi un'interfaccia CommonGoal con un solo metodo. Proponiamo, invece, di eliminare l'interfaccia e di rendere la classe CommonGoalCard astratta e implementare in essa il metodo commonGoalAchieved(Bookshelf bookshelf) che verrà poi overridato dalle sottoclassi, per le quali suggeriamo di utilizzare nomi più significativi, come, ad es., GoalCouples per l'obiettivo delle coppie. Per tale metodo - commonGoalAchieved(Bookshelf bookshelf) - riteniamo sia più facile passare come parametro un Player attraverso il quale si può ottenere la sua bookshelf con un semplice getter, in modo da poter controllare più facilmente se tale giocatore ha già completato l'obiettivo o meno. Suggeriamo infine di non salvare gli scores (8, 6, 4, 2) in una lista, essendo essi già noti.

2.2 Classe PersonalGoal

Salvare una carta obiettivo personale in una matrice 6x5 è solo uno spreco di memoria, in quanto solo 6 caselle su 30 sarebbero occupate (tutte le altre null). Per tale motivo suggeriamo di leggere da file tali carte, scrivendo su 6 righe differenti la combinazione di colore, riga e colonna.

2.3 Enumerazioni

Conviene lasciare solo l'enumerazione più grande, ossia ItemTile, o, ancora meglio, tenere ItemTileType e per la seconda generarne una molto più piccola, ad es. 1, 2, 3. Inoltre, il metodo isEqualType(ItemTile other) è inutile, poichè esiste già il metodo .equals().

2.4 Controller

Abbiamo dei dubbi sui metodi start(), getNextPlayer() e endGame() nella classe MyShelfieGame - nel Model -, che forse sarebbe più corretto spostare nel Controller.

3 Confronto tra le architetture

Le nostre architetture sono molto simili e differiscono quasi solamente per la classe MyShelfieGame nella quale voi gestite il flusso del gioco, compito che noi abbiamo interamente affidato al Controller. Delle vostre feature che abbiamo apprezato nella sezione "Lati positivi" implementeremo il metodo isFull() nella Bookshelf e addNewPlayer() nel Controller.