Peer-Review 1: UML

D'Alessio Edoardo, De Ciechi Samuele, Deidier Simone, Ermacora Iacopo
Gruppo AM-41

Valutazione del diagramma UML delle classi del gruppo AM-04.

1 Lati positivi

- L'implementazione delle personal e common target card con l'aiuto di un file JSON ci sembra un'ottima idea.

2 Lati negativi

- Alcuni aspetti formali dell'UML non vengono rispettati. Per esempio, quando una classe possiede come attributo un'altra classe del diagramma bisognerebbe collegare con una freccia le due classi, indicando alla fine della freccia il nome dell'attributo. Inoltre, mancano tutte le cardinalità nei collegamenti.
- Alcune cose, secondo noi, sono state poste erroneamente nel model anzichè nel controller. Prime su tutte la classe Lobby, che possiede la funzione di gestione e creazione della partita e che quindi dovrebbe appartenere al package controller. In più, a nostro parere, la classe Controller interface è superflua, in quanto la essa potrebbe essere direttamente collegata con User, Shelfie e Lobby.
- Nella descrizione dell'UML viene detto che la personal target card assegnata al giocatore viene salvata all'interno di Shelfie. Nella programmazione ad oggetti questo comporta che Shelfie avrà come attributo la *PersonalGoal-Card* (*sotto forma di freccia di collegamento fra Shelfie e PersonalGoalCard*). Nell'UML, invece, la carta è salvata tramite un Integer, cosa che va contro i principi della programmazione ad oggetti.
- Non è chiaro come avverrà il conteggio dei punti durante la partita: non si capisce quali metodi verranno chiamati, in quale ordine e chi dovrà gestire questo processo.
- In generale nell'UML, manca il passaggio dei parametri nei metodi,che rende meno facile la lettura e l'interpretazione del funzionamento collettivo

del model.

3 Confronto tra le architetture

- Nel gioco le Tile hanno un tipo ed un colore, entrambi attributi che noi avevamo deciso di implementare nel codice. Il tipo ed il colore di una tile però sono sempre in una relazione biunivoca, ossia un certo tipo di tile è sempre associata allo stesso colore. Per questo motivo la scelta di mettere nel codice solamente uno dei due ci sembra migliore e decisamente ottimale.
- Per gestire le caselle della matrice della Board *"superflue"* non viene utilizzata un'altra matrice bitMask (*cosa che avviene nel nostro codice*), bensì vengono aggiunte nuove istanze specifiche nell'enumeration dei Tile. E' una scelta valida ed un'alternativa forse più semplice da gestire il tutto con un'ulteriore matrice.