Peer-Review 1: UML

Davide Capobianco, Leonardo De Clara, Marianna Dragonetti

Gruppo AM52

Valutazione del diagramma UML delle classi del gruppo AM25.

Lati positivi

- I ruoli di ogni classe sono chiari e ben definiti; la logica di gioco è distribuita omogeneamente tra le varie classi, evitando in questo modo la concentrazione di gran parte parte dei metodi in un'unica classe.
- L'uso di interfacce permette una maggiore astrazione e disaccoppiamento tra le classi.
- La presenza di metodi privati di supporto rende possibile una maggiore sicurezza e modularità alle classi che ne fanno uso.

Lati negativi

- Il diagramma UML non risulta molto chiaro e non permette di comprendere a vista d'occhio le relazioni tra le varie classi.
- Non è presente una chiara distinzione tra modalità classica e modalità esperto che, se modellata, potrebbe risultare molto utile per l'uso e la gestione delle carte personaggio.
- L'esistenza di un array per ogni tipo di torre nella classe Island risulta ridondante, soprattutto considerando che in un'isola possono trovarsi nello stesso momento solo torri di un determinato tipo. Una soluzione alternativa potrebbe essere quella di sfruttare l'enum Team all'interno di una struttura dinamica, come ad esempio un ArrayList.
- La modellizzazione del sacchetto come un array di 130 elementi non è molto efficiente in quanto necessita di un ordinamento iniziale: ciò potrebbe essere evitato attraverso la creazione di un array costituito da 5 celle, ciascuna contenente il numero di pedine studente di un determinato colore, oppure attraverso una Hash map che includa le associazioni <Colour, numStudents>.

Confronto tra le architetture

- Ogni classe presenta metodi omogenei ed è in possesso di una identità definita: inoltre, l'esistenza della classe Round, che il nostro gruppo non ha esplicitamente modellato, permette una più equa divisione della logica di gioco.
- L'utilizzo di classi "manager" consente di non esporre interamente agli utilizzatori la struttura delle classi che vengono gestite: è un'opzione che la nostra squadra non ha preso in considerazione.
- La gestione delle carte personaggio attraverso dodici classi distinte concede al modello un alto grado di modularità e scalabilità, nella misura in cui permette l'eventuale modifica del gioco attraverso l'aggiunta di nuove carte, le quali agiranno singolarmente sullo stato dello model. Tuttavia il nostro gruppo considera questa soluzione non ottimale, in quanto riteniamo possibile raggruppare gran parte delle carte in base all'effetto generato.