Peer-Review 1: UML

Elis Kina, Davide Osimo, Alessandro Fusè

Gruppo 38

Valutazione del diagramma UML delle classi del gruppo 39.

Lati positivi

Indicare in questa sezione quali sono secondo voi i lati positivi dell'UML dell'altro gruppo. Se avete qualche difficoltà, provate a simulare il gioco a mano, immaginandovi quali sono le invocazioni di metodo che avvengono in certe situazioni che vi sembrano importanti (ad esempio, la fusione delle isole oppure il calcolo dell'influenza).

Le classi Game, Player e Island sono ben specificate, sia per quanto riguarda i metodi, sia per gli attributi. Le relazioni tra le classi sono corrette sia il tipo sia la cardinalità. Ottima la gestione della plancia di ogni giocatore, con la scomposizione nelle varie "sottoparti" Entrance, Hall, ProfessorsTable e Tower Space.

Lati negativi

Come nella sezione precedente, indicare quali sono secondo voi i lati negativi.

Le carte personaggio sono gestite un po' grossolanamente, il metodo *doAction()* è troppo generico, probabilmente ogni carta ha bisogno di una classe a sé in modo da evitare i metodi che mischiano le carte e ne estraggano tre a caso (quelli nella classe Character Deck).
Alcuni metodi sono descritti nelle classi sbagliate, ad esempio il metodo m*oveTowerToIsland()*, che dovrebbe essere definito da una "GameLogic" dato che non è effettivamente il Player a decidere se spostare o no una torre, è automatico dopo il calcolo della Supremazia; oppure il metodo *moveStudentToCloud()* sempre nella classe Player: un giocatore non ha modo di spostare uno studente su una nuvola, accade il contrario, automaticamente, alla fine di ogni round.

Confronto tra le architetture

Individuate i punti di forza dell'architettura dell'altro gruppo rispetto alla vostra, e quali sono le modifiche che potete fare alla vostra architettura per migliorarla.

Un punto di forza è sicuramente la "densità" delle relazioni, in quantità maggiore rispetto al nostro UML. La gestione dei professori è simile alla nostra, noi non abbiamo utilizzato una classe né abbiamo costruito un *ProfessorsTable* nella plancia di ogni player: ci è sembrato più semplice creare un array all'interno della classe Game al cui interno viene salvato il player "proprietario" del professore associando ad ogni cella dell'array il colore del professore.