Peer-Review 1: UML

Simone Airaghi, Matteo Boido, Lorenzo Bossi

Gruppo 20

Valutazione del diagramma UML delle classi del gruppo 50.

# Lati positivi

* Entrance e DinigRoom sono unite nella DashBoard semplificando le chiamate alle due classi.
* All’inizio della partita hanno pensato alla possibilità di scegliere il lato della tessera nuvola a seconda dei numeri di giocatori della partita.
* Nell’UML si tiene conto del numero di round con un attributo roundNumber nella classe GameControl per sapere quando si è arrivati al decimo round e sono state giocate tutte le carte assistente.
* Design dell’UML ordinato e comprensibile grazie alla divisione in specifici package.

# Lati negativi

* Usare un identificativo per ogni cosa, anche per i singoli studenti, potrebbe essere eccessivo.
* I colori degli studenti potrebbero essere meglio rappresentati da un’enum.
* Non abbiamo compreso come all’inizio del gioco vengano scelte le 3 carte personaggio e di conseguenza come il Player possa capire quali delle 8 carte sono state estratte all’inizio della partita.
* Sarebbe meglio mettere nella StudentBag dei metodi che restituiscano in modo casuale gli studenti.
* Troppa logica applicativa all’interno della classe Player.
* Nell’UML non ci sono metodi che controllino che il Player non giochi una carta già giocata da un altro Player durante la stessa fase di pianificazione.
* Usare un attributo towerColor all’interno della classe Island quando c’è già un attributo Player che ne indica il proprietario è ridondante.
* Non abbiamo compreso come il metodo calulateInfluence della classe Island faccia a calcolare l’influenza prendendo in ingresso una studentList e non conoscendo chi siano i giocatori proprietari dei diversi professori.

# Confronto tra le architetture

* Non avevamo pensato alla possibilità che il giocatore scegliesse il deck.
* Numerare i round per controllare quando finisce la partita potrebbe essere una buona idea.
* Noi abbiamo usato due classi distinte per DiningRoom e ClassRoom, ma potrebbe essere una buona idea unirle in un’unica classe DashBoard per evitare di chiamare ogni volta i metodi getter della DashBoard.