Peer-Review 1: UML

Francesco Tranquillo, Denis Previtali, Andrea Zatti
Gruppo GC37

3 aprile 2022

Valutazione del diagramma UML delle classi del gruppo GC47.

1 Lati positivi

1. L'UML presneta una corretta e completa suddivisione delle classi.
2. Per quanto riguarda la modalità esperto sono stati usati il pattern strategy, per le carte, e il pattern decorator, con la classe TableExpertMode. Questi soddisfano i requisiti richiesti e gli effetti delle carte personaggio.

2 Lati negativi

1. Mancanza di un modo per passare uno o più oggetti alle character-card che permetta di capire la scelta del giocatore, come: l'isola su cui attivare l'effetto, i colori degli studenti, ecc.
2. Abbiamo notato la possibile ridondanza di alcuni metodi come:
   (a) in Bag ci sono addStudent e addStudents che svolgono la stessa funzione, dunque una sola fra le due, indifferentemente, può essere conservata,
   (b) in Island analogamente è inutile avere due metodi addStudent e addStudents,

(c) in DiningRoom il metodo getLine può essere sostituito dal metodo già presente getNumberOfStudentsPerColor, inoltre può essere rimossa la removeStudent(s: Student), il cui compito è già svolto da removeStudent(c: Color).

3. In Entrance, per quanto riguarda il metodo removeStudent(student: Student), non avendo all'esterno i puntatori degli studenti (interni) non è possibile passare, come attributo, lo studente da rimuovere.

4. In Island è meglio non avere un puntatore ad una singola torre, a meno che non si metta lo stesso id a più isole durante l'unificazione, che però è contraddittorio rispetto al concetto di identificatore; questo rende ambiguo il contatore numOfTowers che nell'eventualità in cui dovesse assumere come valore il numero di isole unificate sarebbe ridondante, se invece dovesse assumere solo i valori 0 o 1 potrebbe essere sostituito da un boolean.

5. Per quanto riguarda Player, l'attributo towerColor concettualmente sarebbe meglio metterlo in Towers. Inoltre all'esterno di Player non si hanno collegamenti con la classe Assistant, per questo bisognerebbe aggiungere metodi in Player che ne permettano l'accesso.

6. In Table non abbiamo trovato buoni motivi per avere lista di studenti visto che sono già su Cloud, Dashboard o Island.

7. Per come è stato strutturato l'UML il controller dovrà collegarsi anche a molte altre classi come DiningRoom, Entrance, Island ... per chiamarne i metodi; sarebbe meglio implementare in Table dei metodi che li chiamano a loro volta.

8. Infine, un possibile errore di distrazione è dato dalla presenza di due frecce con cardinalità differenti tra Table e Player, oltre che tra Player e Assistant.

3 Assunzioni personali

1. Supponiamo esistano metodi di get e set non esplicitati, come ad esempio nella classe MotherNature della quale altrimenti non sarebbe possibile scoprirne e modificarne la posizione.

4 Confronto tra le architetture

Ci siamo accorti che nel nostro UML mancava un modo per reinserire gli studenti nella Bag (tramite l'effetto della carta personaggio numero 12), oltre che a un modo per rimuovere gli studenti della DiningRoom (a causa dell'effetto della carta personaggio numero 10). Inoltre abbiamo notato due metodi funzionali, anche se non necessari per come avevamo pensato e costruito l'UML, che sono getPlayerWithMaxProfessors() e getPlayerWithMinTowers(). Infine poiché abbiamo notato la mancanza di cardinalità e direzionalità di alcuni dei collegamenti delle classi nel nostro UML provvederemo ad aggiungerli.