Peer-Review 1: UML

Emanuele Paci, Lorenzo Paleari, Thomas Puppinato
Gruppo 12

4 aprile 2022

Valutazione del diagramma UML delle classi del gruppo 22.

1 Lati positivi

Analizzando l'UML abbiamo individuato alcune caratteristiche degne di nota, quali la presenza di un controller diviso in più handler, l'utilizzo del pattern Decorator, dello Strategy pattern e la gestione delle carte personaggio nel Model.

L'aver suddiviso il controller in più handler facilita la comprensione immediata del modello, rendendo anche più semplici eventuali modifiche future oltre a rendere il tutto più organizzato e lineare.

L'idea di sfruttare uno Strategy pattern per gestire l'attivazione di quei personaggi che modificano le regole del gioco (cambiamenti sul modo di calcolare l'influenza, controllo dei professori...) è sicuramente brillante e funzionale.

Infine, apprezzabile anche il non aver creato una classe per ogni personaggio ma essere riusciti a sfruttare le caratteristiche comuni di ogni carta, si nota in particolare la distinzione tra le carte con studenti al loro interno e quelle che non li hanno.

2 Lati negativi

Secondo noi questo UML è molto ad alto livello, andrebbero specificati più metodi per comprendere meglio come si ha intenzione di gestire una partita. Sarebbe stato meglio descrivere visivamente lo Strategy Pattern presente, anziché farlo evincere da una nota a margine, in maniera tra l'altro non del tutto chiara: lo Strategy infatti necessita di più classi per funzionare correttamente, cosa che non osserviamo all'interno del modello.

Risulta poco chiaro in che maniera andranno ad essere create le carte personaggio, nonché come vengano "legate" al loro effetto e anche come si intende far attivare gli

effetti delle carte personaggio.
Non sembra esserci un modo per evincere dall'effetto a quale oggetto CharacterCard fa riferimento e viceversa, il che potrebbe presentare un problema a livello di implementazione.
L'implementazione effettiva degli effetti potrebbe rivelarsi complessa, osservando l'UML sembra esserci il rischio di incorrere nell'utilizzo di grossi if statement per il corretto indirizzamento del metodo playEffect().

Veramente di difficile comprensione la struttura della classe board della quale non capiamo come vengono gestiti gli studenti, sembrerebbero mancare delle indicazioni che chiariscano la vostra idea, abbiamo inoltre notato che avete usato un vettore di booleani per rappresentare la presenza o meno dei professori, personalmente pensiamo che l'utilizzo dei colori renderebbe più semplice l'implementazione e anche l'immediatezza della classe.

3 Confronto tra le architetture

A differenza del nostro UML, la parte di model ci sembra molto più scarna e contiene l'essenziale per mantenere lo stato della partita, una parte di esso rappresentata dalla classe game è stata messa nel controller e svolge le funzioni di quest'ultimo, invece noi abbiamo preferito separare la classe game, tenendola nel model, dalla classe controller.

Inizialmente anche noi avevamo optato per raggruppare tutti i personaggi in una sola classe generica nel Model e avevamo pensato di implementare gli effetti delle carte tramite un pattern decorator nel controller, similmente a quanto fatto da voi.
Abbiamo successivamente cambiato idea in quanto abbiamo riscontrato problemi e difficoltà a livello implementativo, optando invece per l'utilizzo di una singola classe che gestisse gli effetti, situata nel controller, e qualche classe nel Model che mantenesse sia lo stato che l'effetto dei personaggi stessi.

Similmente a quanto fatto da voi, anche noi abbiamo optato per utilizzare uno Strategy pattern, così come la realizzazione dei player con i relativi deck e per finire è pressoché identica l'idea di associare alle isole un intero che ne indichi la dimensione.

Prendendo spunto da voi, invece, potremmo migliorare il nostro design andando a suddividere la classe controller in più handler per facilitare l'implementazione, la chiarezza e la leggibilità.