Peer-Review 1: UML

Moretti Lorenzo, Mormile Matteo, Orofino Antonino
Gruppo 6

3 aprile 2022

Valutazione del diagramma UML delle classi del gruppo 5.

1 Lati positivi

Riteniamo che il design generale del Model sia stato ben strutturato. Da una prima analisi, infatti, il diagramma mostra classi non troppo cariche di metodi e di conseguenza una corretta applicazione del principio di suddivisione delle responsabilità. La gestione delle isole, in particolare, ci è sembrata abbastanza conveniente. Grazie all'aggiunta dell'attributo `size` all'interno di `Island` è stato rimosso il problema della gestione di un *arcipelago* di isole.

Per quanto riguarda le classi `Cloud`, `Bag`, `Player` e le carte, crediamo che tutte quante riescano a convogliare gli aspetti più significativi e necessari allo svolgimento del gioco.

2 Lati negativi

Innanzitutto riteniamo che siano stati tralasciati quasi tutti gli attributi necessari per rappresentare le relazioni di contenimento tra classi. Per fare un esempio, `Game` non contiene nessun attributo di tipo `Table` per memorizzare la plancia del gioco.

`Player` e classi annesse

Non siamo riusciti a capire il ruolo della classe enumerativa `Mages` e di conseguenza, il significato dell'attributo di questo tipo all'interno di un giocatore.

Non siamo molto d'accordo sulla scelta di memorizzare i professori all'interno di ogni singola plancia. Se la logica viene mal gestita, è più probabile il verificarsi di situazioni in cui lo stesso professore risulti posseduto da più giocatori contemporaneamente.

Riteniamo inoltre che non sia necessario mantenere un riferimento a `Table` all'interno di `Plance_scuola`. In questo modo sia il gioco che ciascuna plancia posseggono un riferimento allo stesso oggetto.

Come ultima osservazione, notiamo che manca l'attributo che permetta di capire a chi appartiene un determinato colore di `tower` ed un riferimento all'ultima carta assistente giocata. Risulta impossibile, infatti, risalire al giocatore *proprietario* delle torri di una determinata isola ed alla sua ultima carta giocata.

`CharacterCard` e classi direttamente collegate

Crediamo che l'utilizzo di una classe `Posto_carte` sia superfluo. Essa contiene soltanto un array come attributo e nessun metodo.

Riteniamo azzeccata la scelta del design pattern *Strategy* per gestire il comportamento di ciascuna carta personaggio. Tuttavia, l'applicazione risulta errata. L'interfaccia `CharacterBehavior` possiede un unico metodo `useCharacter` che non richiede parametri, eppure quest'ultimo non viene implementato da nessuna delle classi personaggio. Il metodo risulta implementato con segnatura diversa in ogni classe.

`Student` e `StudentType` Pensiamo che la classe `Student` possa essere eliminata in favore della sola classe `StudentType`. Inoltre, crediamo possa essere opportuno aggiungere un parametro di tipo intero all'interno della classe enumerativa. Questo parametro potrebbe rappresentare l'indice all'interno degli array usati per memorizzare gli studenti in diverse classi. In questo modo si evita il problema di dover ricordare l'ordine delle creature magiche in questi vettori.

`Cloud`
Mancano i metodi per inserire e rimuovere gli studenti dalle nuvole.

3 Confronto tra le architetture

Riteniamo più che valida la scelta del pattern *Strategy* per modellare il comportamento delle carte speciali personaggio. Anche nel nostro progetto si è optato per una stessa gestione, tuttavia, abbiamo preferito spostare questa logica lato controller.

Per quanto riguarda la struttura dati utilizzata per memorizzare gli studenti e gli insegnati non siamo molto felici della scelta di un semplice array. L'utilizzo di un vettore richiede, infatti, la scelta e la memorizzazione di un ordine delle 5 creature magiche. Nel nostro modello abbiamo preferito utilizzare una mappa che utilizza come chiavi i valori enumerativi del tipo di creatura e come valori degli interi che rappresentano il numero di studenti appartenenti ad una famiglia di creature. In questo modo non abbiamo il problema di dover ricordare l'indice, all'interno dell'array, di una determinata creatura.

Per quanto riguarda la gestione delle isole e della loro unione, abbiamo avuto la stessa idea. Sia noi che loro, infatti, abbiamo deciso di eliminare le isole a seguito di unione e di aumentarne il valore dimensione per rappresentare un isola *Arcipelago*.

Un'ultima differenza che vogliamo far notare, è la scelta di dove posizione i professori. Se da un lato il gruppo 5 ha deciso di memorizzare questa informazione all'interno della plancia di ciascun personaggio, noi abbiamo ritenuto più opportuno la creazione di una classe helper che si occupasse della loro gestione. Utilizzando questo approccio abbiamo eliminato la possibilità che lo stesso professore, per errore, possa risultare in possesso di più personaggi allo stesso tempo.