Peer-Review 1: UML

Claudio Arione, Riccardo Begliomini, Giuseppe Boccia

Gruppo AM17

Valutazione del diagramma UML delle classi del gruppo AM47.

Lati positivi

Dopo aver analizzato con attenzione l'UML fornitoci dal gruppo revisionato, abbiamo evidenziato i seguenti aspetti positivi:

- gestione di studenti e professori: i due oggetti di gioco sono resi tramite la stessa classe *TokenCollection*, facilitandone la gestione e rendendo il codice più riutilizzabile;
- gestione delle partite multiple: attraverso la classe *GameRepo* è possibile istanziare partite multiple in modo lineare e comprensibile, associando a ogni *Game* un ID;
- gestione della partita a quattro giocatori attraverso gli attributi *Squad* e *TowersColour*, a cui ogni giocatore è associato;
- gestione degli argomenti per le carte personaggio: la classe *CharacterCard*, infatti, usa una *Map<String, Object>* per passare gli argomenti dell'effetto da utilizzare. Questo approccio fa sì che il metodo non debba avere una lunga lista di parametri.

Lati negativi

Nonostante il progetto sia nel suo complesso ben strutturato, abbiamo osservato alcuni aspetti che a nostro parere possono essere migliorati:

- fusioni tra le isole: è presente un attributo *size* ma non è ben chiaro come vengono gestite le fusioni tra le isole. Viene creata un'altra istanza di *Island* oppure si aumenta la size di una, eliminando le altre?
- Gestione delle carte personaggio e dei relativi modificatori di turno: per esempio, come faccio a sapere se l'influenza su un'isola deve ignorare le torri oppure un dato colore?
- *TowersColour* e *Squad* sembrano gestire la stessa informazione: che differenza c'è tra le due?

Confronto tra le architetture

Considerando i sopracitati lati positivi e negativi, e dopo aver confrontato l'architettura del gruppo revisionato con la nostra, riportiamo qui di seguito un breve confronto tra le scelte implementative dei due gruppi:

- l'architettura del gruppo revisionato presenta un oggetto *Command*, volto a modificare lo *State* del *Game*, mentre la nostra architettura ha dei metodi che modificano gli attributi degli oggetti di gioco. Ciò indica che la loro architettura è più orientata ad una futura integrazione con il controller di quanto la nostra lo sia al momento.
- Studenti e professori sono stati implementati all'interno della stessa classe *Token* e di essi viene memorizzato solamente il numero, a differenza della nostra scelta di usare degli oggetti. Dopo un'attenta analisi, abbiamo deciso di mantenere i nostri *Student* come oggetti, ma di implementare i professori con una *Map<Color, Player>*, dove il *Player* indica chi possiede il professore di quel *Color*.
- Programmiamo di rivedere il modo in cui gestire il possesso delle isole per implementare le partite a quattro giocatori;

- Mentre noi abbiamo diviso *Entrance* e *DiningRoom* in due classi separate, l'altro gruppo ha deciso di avere nella *Dashboard* due istanze della stessa classe *TokenCollection*.