Peer-Review 1: UML

Andrea Pinessi, Martina Riva, Martina Quaregna, Alessandro Vai

Gruppo GC24

Valutazione del diagramma UML delle classi del Gruppo GC14.

Lati positivi

- La suddivisione in tre le diverse posizioni della plancia (in base al numero di giocatori);
- L'utilizzo di un metodo in LivingRoom che controlli i bordi delle tessere;
- Utilizzare un'ArrayList come classifica in **CommonGoal** per assegnare i punti;

Lati negativi

1. L'UML in gran parte dei casi **non** rispetta le regole sintattiche e formali:
    a. attributo: tipo
    b. metodo(): tipo restituito

    ➔ **OSSERVAZIONE:** metodi e attributi devono iniziare con una lettera minuscola
2. Sono presenti due classi "Abstract" (AbstractLivingRoom e CommonGoal):
    - in **AbstractLivingRoom** ci sono due attributi *protected*:
        - **Card Plank** e **boardmap:** Card Plank impedisce la modifica in partita della plancia (estrazione delle tessere, ...), è un problema;
    - **AbstractLivingRoom** possiede un attributo che non può essere modificato, di fatto risulta "unused" (**totalCardsBag** può essere modificata unicamente da **FillLivingRoom**).
    - In **CommonGoal** sono stati implementate 9 classi che controllano le Common Goal Card, i metodi di alcune di queste (#1, #3, #5, #6, #7, #9) sono poco coerenti e coesi nella scelta dei parametri passati ai metodi, risultano pertanto confusi e poco leggibili.
3. L'implementazione della **MappingNodeBoard** risulta abbastanza scomoda
4. L'attributo **okToDraw** ha un problema:
    a. Non è presente un metodo per poterlo modificare (il metodo **CheckBorders** non può modificare questo attributo poiché privato)
5. Il fatto di avere l'attributo **online** può essere una buona idea ma presenta un paio di problemi:
    a. Nel momento in cui un giocatore viene settato come "online" (e viene quindi abilitato al turno) non ho la certezza che il giocatore sia ancora connesso dopo il set di questo attributo; di conseguenza potrei erroneamente avere un giocatore "online" che però non è più in questo stato;
    b. Questo attributo dovrebbe essere aggiornato continuamente, sarebbe meglio controllare lo stato del giocatore nel server.
6. Non avere una classe "bookshelf" è un problema, infatti l'inserimento delle tessere nella libreria è molto più complesso da gestire
7. Infine, il calcolo del punteggio, per come si presenta questo UML può essere effettuato esclusivamente a fine partita

Confronto tra le architetture

Differenze sostanziali

- Nell'UML del gruppo 24 sono presenti classi assenti nell'UML del gruppo 14 (ad esempio **Bookshelf**, **ScoringToken**) mentre nell'UML del gruppo 14 questi due sono stati implementati all'interno di altre classi come attributi
    - A nostro avviso il fatto di avere queste due classi in più rappresenta un vantaggio, questo perché il codice risulta più modulare e molto più facilmente controllabile e aggiornabile in futuro.
- Nell'UML del gruppo 14 sono presenti tre diverse inizializzazioni della plancia a inizio partita (in base al numero di giocatori in partita), questo tipo di implementazione è comoda perché permette di separare le tre conformazioni e di poterle generare all'occorrenza.
- Nell'UML del gruppo 24 (a differenza del gruppo 14) è stato implementato lo *Strategy Pattern* per quanto riguarda le **CommonGoalCard**, questo ci permette di gestire in maniera più sistematica il controllo sulla libreria.
- Nell'UML del gruppo 14 è assente la caratterizzazione di una chat, cosa presente nell'UML del gruppo 24.

Similitudini

- Entrambi i gruppi hanno interpretato come classi fondamentali le due classi:
    - **Player** e **Game** [Gruppo 24]
    - **Player** e **MatchStatus** [Gruppo 14]

  Questo tipo di gestione è in effetti molto efficace perché permette di concentrarsi sulle due entità fondamentali che compongono la partita.
- Entrambi i gruppi hanno optato per l'utilizzo di file JSON per la gestione delle **PersonalGoalCards**, questo soprattutto perché risulta estremamente comodo avere 12 oggetti immutabili che, all'occorrenza, possono essere letti da questi file.