Peer-Review 1: UML

Alice Calcaterra
Nicolas Carboni
Andrea Cavezzale
Daniel Ciobanu
Gruppo AM-17

Consegna del 03/04/2023

Valutazione del diagramma UML delle classi del gruppo AM-26.

1 Lati positivi

Già dal primo impatto l'UML si presenta chiaro e organizzato nel suo schema generale.
Le classi sono ben distinte, ognuna ha i propri metodi e attributi da cui risulta immediato capire quale siano le varie funzioni.
Lo schema è modulare, la separazione dei ruoli svolti da ogni oggetto è ben definita, ogni componente di gioco trova un corrispettivo nell'UML (Bookshelf, CommonCards, PersonalGoalCards...) e sono ben utilizzate delle classi di supporto come DataCard e Coordinates. Un altro elemento positivo da sottolineare è l'implementazione delle Bags (BagItem, BagCommon e BagPersonal), esse infatti contribuiscono all'ordine e a rafforzare la corrispondenza tra oggetti e funzioni di gioco.

Lato controller, nonostante l'UML non presenti una struttura particolarmente articolata, si osserva come si è già pensato ad una possibile gestione della comunicazione con il client attraverso la classe "communicationHandlerSocket" e la classe "Message".
Sono osservabili inoltre dei metodi per l'invio e ricezione di messaggi fra client e server in diverse varianti (Send, SendAllExcept...).
Infine, spicca l'aver già pensato all'introduzione della funzionalità di chat (anche se per ora solo abbozzata) attraverso il metodo addChatMessage e la classe "Chat".

Dal diagramma si osservano anche dei buoni elementi di ottimizzazione e design, a partire dall'accorpamento di alcune CommonCards: si rinuncia a rappresentare degli oggetti di gioco 1:1, a guadagno di un riutilizzo delle funzioni.

Si nota inoltre un buon utilizzo del pattern AbstractFactory per la creazione delle CommonCards.

2 Lati negativi

Il principale aspetto negativo che è stato riscontrato, è la poco chiarezza tra le interazioni tra Model e Controller, essa infatti non risulta molto comprensibile dallo schema. E' stata notata la classe PlayerIterator, ma non è ben chiaro come si applichino le altre funzioni di gioco.
A questo proposito si consiglia l'implementazione di alcune interfacce nel Model che definiscano in modo più chiaro come il Controller possa relazionarsi con esso.

Continuando su questa linea, ci sono state delle difficoltà nel capire il flusso di gioco; esso è in generale difficilmente inferibile da un diagramma, ma una breve descrizione nella relazione avrebbe aiutato nella comprensione.

Nella parte dedicata alla comunicazione si osserva la classe "ModelObserver".
Si suggerisce, in questo contesto, la sostituzione con un ModelListener come visibile nei materiali utilizzati durante la seconda esercitazione relativa al progetto (Pattern MVC).

Visti questi tre elementi principali, ci sono altri due aspetti di minore rilevanza su cui ci si vuole soffermare:

- Si consiglia per la classe ItemType di utilizzare direttamente l'enumerazione dei colori, senza far uso dei numeri interi che risultano superflui;
- Inoltre considerando la classe Item, dal diagramma risulta non necessario che gli oggetti abbiano un riferimento coordinates alla Bookshelf, quando già quest'ultima ha una matrice di Item negli attributi. Essendo questo ridondante, si consiglia di modificare la classe Item, e renderli degli oggetti "passivi".

3 Confronto tra le architetture

L'architettura analizzata presenta molte similitudini rispetto a quella realizzata da noi, soprattutto nella struttura generale: la distinzione delle classi e dei ruoli è praticamente coincidente con la nostra, e i metodi e gli attributi sono pressoché identici.
Si osservano alcune differenze nell'approccio al pattern MVC, il nostro gruppo ha preferito l'utilizzo di interfacce fra Model-Controller (in maniera tale da porre in evidenza i metodi che il controller potrà invocare) e l'utilizzo di Listeners al posto di Observers nella comunicazione Controller-Model e Model-View.

Dallo schema dell'altro gruppo spicca una gestione ottimizzata delle CommonCards, che sfrutta vari pattern. Le nostre scelte implementative a questo proposito sono leggermente diverse: vengono accorpati obiettivi simili mediante l'invocazione degli stessi metodi e algoritmi, ma comunque per ognuna delle 12 CommonCards è presenta la relativa classe.

Due aspetti particolarmente rilevanti da cui si vuole prendere spunto sono i seguenti: l'implementazione della "regola di prima partita" sfruttando l'enum GameMode; l'utilizzo delle varie Bags per una gestione ottimale delle CommonCard, PersonalCards e gli Item.

Concludiamo dicendo che rimaniamo a disposizione per eventuali chiarimenti sui nostri commenti, in modo da avere un ulteriore confronto e un dialogo, sia per via scritta sia di persona.