Peer-Review 1: UML

Jonel Relucio, Dalila Samr, Alessandro Petruzzelli, Lucian Sas

Gruppo 25

Valutazione del diagramma UML delle classi del gruppo 15.

Lati positivi

- È stata individuata una buona suddivisione delle classi ed è stato mantenuto il principio di singola responsabilità
- La classe “Game” è utile per avere una visione d’insieme della partita e risulta molto comoda nel caso di decidesse di implementare la FA “Persistenza”
- La classe astratta “CommonGoal” migliora la leggibilità e la manutenibilità del codice, in quanto, volendo in futuro inserire una nuova carta, basterà aggiungere una nuova classe, invece di ricorrere alla modifica di un eventuale switch-case molto complesso da gestire

Lati negativi

- Si potrebbero raggruppare alcune common goal card che presentano una logica di ricerca simile sotto un’unica classe astratta o interfaccia, in modo tale da non dover implementare 12 algoritmi diversi per verificare il raggiungimento di un obbiettivo comune, migliorando l’estensibilità nel caso in futuro si volesse aggiungere una nuova carta con logica simile alle altre.
- Nell’UML aggiungere i nomi ai parametri dei metodi, utilizzare la nomenclatura inglese (presa dalle regole in inglese) e aggiungere i riferimenti mancanti all’interno delle classi, il tutto per migliorare la leggibilità del diagramma UML.
- Il metodo “calculatePoints()” definito nella classe “Game” secondo noi dovrebbe essere parte di un controller, così come la logica che verifica se una tessera possa essere prelevata dalla board.
- Manca un riferimento al “Player” corrente e alla lista dei giocatori all’interno della classe game, inoltre non è chiaro come venga gestita la turnazione all’interno del gioco.

Confronto tra le architetture

Le architetture sono tutto sommato molto simili tra di loro, inclusa la classe “Game” che mantiene lo stato generale della partita.

La classe “Board” del gruppo 15 è rappresentata mediante una mappa che ha come chiave un oggetto di tipo coordinata, e come valore la carta oggetto presente alla relativa coordinata, mentre la “Board” del nostro gruppo è semplicemente un array bidimensionale di “ObjectCard”.

L’utilizzo di un tipo complesso come una Mappa potrebbe essere utilie nell’implementazione dei vari algoritmi in quanto sarà possibile utilizzare tutti i metodi messi a disposizione dalla classe Map di Java ed utilizzare per esempio la programmazione funzionale per snellire il codice.
Potremmo utilizzare un approccio simile per semplificare l’implementazione dei nostri algoritmi un modo tale da renderli più leggilbili.

L’utilizzo di una classe a parte per identificare le coordinate potrebbe essere un vantaggio in futuro se si decidesse di inserire più tipologie di “Board”, in quanto si potrebbe estendere la classe “Board” e la classe “Corodinate”, e comporre ciascuna “Board” con il relativo tipo di “Coordinate”