Peer-Review 1: UML

Luca Pagano, Fabio Sabbion, Andri Salillari, Lorenzo Torsani

Gruppo GC09

Valutazione del diagramma UML delle classi del gruppo GC49.

Lati positivi

- *PersonalGoalCard*: molto intelligente la soluzione di avere una *Coordinates* per ogni tipologia di tessera; potrebbe essere ulteriormente perfezionata sostituendolo con una mappa di Category e Coordinates, così da aumentare l'estendibilità. Se in futuro si volesse aggiungere un'ulteriore tipologia, si dovrebbe modificare la classe aggiungendo un ulteriore attributo, con relativo getter.
- *Pocket:* soluzione elegante avere una classe che gestisca le Tiles, rendendo indipendente la logica di prelievo.
- *Score* facile da ottenere; non è necessario ricalcolarlo ogni volta, alleggerendo la computazione.

Lati negativi

- In generale i nomi non rispettano le convenzioni. Per esempio, nei getter inserire *get* prima del metodo (vedi *PersonalGoalCard*), *ConditionCheck* è un metodo che inizia con la maiuscola e alcuni nomi potrebbero essere resi più leggibili scrivendoli per esteso.
- Per quanto riguarda *image* in *TileType,* sarebbe meglio inserirla nel costruttore come *final*, in quanto la variante dell'immagine della Tile non dovrebbe essere modificata.
- *CommonGoalCards*: non facilmente estendibile in quanto è stata implementata una classe per ogni tipologia di CommonGoal; se si volesse aggiungere un ulteriore tipo si dovrebe modificare l'intera classe. Oltretutto, non esiste un metodo per crearle in automatico. Inoltre, non viene rispettato il paradigma ad oggetti nella distinzione fra le varie tipologie all'interno delle singole classi (es: *isEleven*, *numToLook*); sarebbe stato più coerente avere una classe astratta, estesa successivamente dalle singole implementazioni.
  In *getAchieved*, inoltre, la gestione delle *CommonGoalCards* completate dal singolo giocatore non è facilmente estendibile, essendo implementata con un *Pair*. Nel caso in futuro il numero di CommonGoalCards dovesse essere maggiore di due, si dovrebbe modificare il tipo dell'attributo.

Confronto tra le architetture

- Per il controllo dell'obbiettivo delle *CommonGoalCards* anche noi ci serviamo di uno *strategy pattern*; differentemente da questa implementazione, abbiamo implementato la possibilità di costruire la CommonGoalCard da un JSON, migliorando l'estendibilità.
- *Score* aggrega i punteggi mentre noi abbiamo deciso di mantenerli separati i punteggi nel caso provenissero da obbiettivi diversi, come nel gioco da tavolo, e li calcoliamo solamente alla fine.
- La variante dell'immagine della Tile viene gestita con un *int* mentre noi abbiamo deciso di implementare un ENUM
- Il metodo *getTile* rimuove le tessere dalla board una alla volta. Per realizzare ciò, l'attributo *pickable* gestisce la possibilità di selezionare una Tile dalla board*,* il quale viene aggiornato nel caso ci siano Tile adiacenti o meno. La nostra implementazione consiste invece nel

selezionare e rimuovere più tessere contemporaneamente, e il controllo delle Tile adiacenti viene realizzato in modo integrato.