Peer-Review 1: UML

Filippo Balzarini, Matteo Boglioni, Michele Cavicchioli, Christian Biffi
Gruppo 07

3 aprile 2023

Valutazione del diagramma UML delle classi del gruppo 16.

1 Lati positivi

1. In bookshelf è interessante l'overloading del metodo placeTiles che rende piuttosto rigido e perciò safe l'inserimento delle Tiles che possono essere appunto da una a tre. L'unica perplessità è rispetto a come questi metodi possano essere invocati senza usare una serie di "if" in cascata.

2. Divisione delle CommonGoalCard in modo che i controlli possano essere specifici per ogni tipo di carta, positivo il fatto che si sia pensato di raggruppare pattern analoghi. Sarebbe stato meglio indicare esattamente quali fossero i raggruppamenti individuati (CommonGoalCard# è poco chiaro).

3. Il vincolo della direzione sulla scelta di una tile rispetto alla precendente dalla living room board è un'ottima idea. Ciò permette di semplificare la logica per il controllo della validità della mossa.

2 Lati negativi

1. All'interno della classe Game i metodi getPlayers(), getCurrentPlayer() restituiscono un oggetto /ArrayList<> di Player al chiamante. La scelta sembra poco safe, esporre tale classe infatti permette dall'esterno di avere accesso ai metodi modificatori.

2. Per quello che si può dedurre dall'analisi dell'UML, l'aggiunta di TurnController è utile solo a scopo logico e non porta abbastanza vantaggi per giustificare l'aggiunta di complessità a livello implementativo. Basterebbe un solo controller che concentra le azioni di entrambe le classi.

3. Gli attributi x e y della classe Cell risultano inutili dato che la cella stessa è inserita in una matrice e per ispezionarla è richiesto di conoscere le coordinate in modo pregresso. Avere Cell con tali attributi richiede inoltre di aggiornare le coordinate x e y coerentemente alla posizione nella matrice quando tale Cella viene spostata da Board a Bookshelf, non ottenendo tuttavia alcun vantaggio.

4. La suddivisione delle CommonGoalCard in sottoclassi non è sfruttabile se i metodi checkScheme hanno signature differenti. Non è possibile infatti in questo caso sfruttare il pattern di tipo strategy dal momento che il chiamante deve già conoscere la signature dello specifico checkScheme chiamato e perciò il tipo dinamico della CommonGoalCard in questione.

5. Abbiamo notato il metodo isPickable(x,y) nella classe Board. Il suo significato è ovvio, ma a livello di UML non si distingue in alcun modo una cella non utilizzabile (es: 00) rispetto ad un'altra che invece è occupabile da tiles durante il gioco (es: 44); questo porta il metodo isPickable ad essere scritto in hardcode. Un consiglio è quello di aggiungere un attributo per distinguere il "tipo" di cella.

3 Confronto tra le architetture

A valle dell'analisi fatta sull'UML e sul documento esplicativo allegato abbiamo ritenuto che le nostre scelte di design nel complesso garantiscano maggiore sicurezza e robustezza. Le somiglianze in ogni caso non mancano, soprattutto nella parte di CommonGoalCard. Ci sono comunque nelle due architetture delle differenze; un esempio è la gestione delle celle tra living room board e bookshelf, con il gruppo revisionato che sfrutta una classe Cell comune per entrambi gli oggetti, mentre noi implementiamo questo concetto solo sulla living room board tramite degli Slot, usando poi una semplice matrice di tiles per rappresentare la bookshelf. Un'ulteriore diversità la si trova nel fatto che

il gruppo revisionato rappresenta tramite una enum la direzione di scelta di una tile dalla living room board rispetto alla precedente (Lato Positivo n.3), rendendo così più efficiente e semplice il controllo sulla validità della mossa, dall'altro lato però sarà inevitabile l'appesantimento dei controlli nel client, un consiglio che ci sentiamo di dare è quello di rappresentare anche le scelte diagonali, così da semplificare i check locali, altrimenti come vengono rappresentate le eventuali scelte diagonali? Noi abbiamo invece deciso di lasciare un po' più di libertà di scelta al client per poi verificare più tardi la validità della mossa, quindi senza la rappresentazione di direzionalità; entrambe le scelte di design sono più che valide, ci sono vantaggi e svantaggi in entrambi i casi.