Peer-Review 1: UML

<Agostino Contemi >, <Federico Consorte>, < Filippo Dodi>, <Jacopo Corsi>

Gruppo <04 >

Valutazione del diagramma UML delle classi del gruppo < 13>.

Lati positivi

Abbiamo riscontrato una serie di lati positivi durante l'analisi dell'UML del gruppo assegnatoci. In particolare abbiamo apprezzato l'utilizzo delle interfacce per la gestione dei punti in quanto separa i PointsPlayable dai PointsObjective che vengono calcolati in due momenti differenti della partita.
Nella classe Field è comodo il metodo per trovare le posizioni di ciascuna carta di modo da tenere traccia dell'evoluzione di ciascun field.

Lati negativi

Alla luce di ciò che abbiamo compreso dal vostro UML abbiamo riscontrato delle criticità che volevamo condividere con voi. Vista la mancata implementazione delle classi CardResource, CardStarter e CardGold il lavoro risulta incompleto e non facilmente comprensibile. Le interfacce CardPlayableIF, CardObjectiveIF e CardStarterIF devono essere implementate e non viceversa. Nella classe Match dove è presente Player va sostituito con PlayerIF. Potrebbe essere utile creare una CardSidePlayableIF, dare dei metodi a CardSide e sovrascriverli alle sottoclassi.
Nella enum dei colori il black non è necessario in quanto il FirstPlayer è il black. Non comprendiamo l'utilità della classe token, potrebbe esserci invece un riferimento al colore all'interno del player .

Confronto tra le architetture

Prendendo spunto dal vostro utilizzo delle coordinate ci siamo resi conto che nel nostro manoscritto non avevamo la possibilità di distinguere una carta piazzata sotto e una carta piazzata sopra, problema che abbiamo risolto aggiungendo l'attributo PlacementTurn in CornerCardFace.
Le architetture risultano estremamente differenti, a partire dalla scelta di inserire la logica di gioco interamente nel GameModel lasciando al Controller le funzioni di inoltro dati. Abbiamo apprezzato l'utilizzo della card factory, infatti, anche noi riteniamo sia il design pattern adeguato per la creazione di tipi differenti di carte o mazzi da gioco, noi invece abbiamo utilizzato il metodo CreateCards nell'interfaccia deck che poi viene implementato in modo diverso a seconda della classe che implementa tale interfaccia (override).