Peer-Review 1: UML

<Michele Adorni>, <Edoardo Carlotto>, <Giacomo Brunetta>, <Gianluca Cantonetti>

Gruppo <AM09>

Valutazione del diagramma UML delle classi del gruppo <AM18>.

Lati positivi

L'UML, secondo noi, è molto buono. Ci sono scelte di design ben fatte.

Il controller è ben fatto, la classe turnController per gestire il turno corrente ci sembra una buona scelta di design.

Ci sembra un ottimo metodo l' "updateTakeable()", è effettivamente necessario un metodo come questo per evitare bug e problemi anche con la GUI.

I metodi della board sono ben dichiarati e tutti estremamente utili. Sembrano compiere tutti i lavori che deve svolgere la board, e questa è un'ottima cosa.

La classe Tile è ok (anche se, come scritto nei "lati negativi", secondo noi sarebbe stato meglio fare una enum), ha le caratteristiche adeguate per essere lavorata nella view.

Il player è ben fatto e ci piacciono i metodi per aggiornare e controllare il punteggio, sono molto chiari.

Anche la bookShelf è ben fatta, ci sono tanti metodi, ma sembrano tutti necessari.

Lati negativi

La board può essere fatta con il passaggio di un Json, sarebbe sicuramente più snella ed elegante. Questo porta a dover creare quattro sottoclassi board[numPlayers]P che, secondo noi, appesantisce il codice. Mediante il passaggio di un Json, basterebbe una sola classe Board che riceve un Json e lo gestisce (per quanto riguarda la differente Board a seconda dei numPlayers si può direttamente lavorare nel Json, senza dover dichiarare quattro sottoclassi).

Le dodici classi per i CommonGoals sono una scelta architetturale, secondo noi appesantiscono un po' il codice e l'UML, ma come già detto, sono una scelta propria di design che, secondo noi, non è definibile come un vero e proprio lato negativo; si sarebbe potuta fare più leggera, ma anche così fa il suo lavoro.

Per quanto riguarda le tile, secondo noi sarebbe stato meglio fare una enum direttamente per le Tile e non per i colori. Ci spieghiamo meglio: si poteva fare una enum per le Tile (definendo magari Invalid) e poi una "sotto"-enum per i colori che veniva passata alla enum principale, in modo tale da non dover definire una classe per le tiles.

Perché non unire i metodi "initializeGenericBoard()" e "initialize"?

Secondo noi è necessario un metodo “getPlayers()” che ritorni la lista dei players nel Game.

Confronto tra le architetture

La nostra architettura, per certi punti di vista, è più snella. C’è qualche metodo più pesante da noi, che potrebbe essere alleggerito come fatto da voi, aggiungendo qualche metodo in più.

Ad esempio, troviamo molto eleganti i metodi del Player sul punteggio, rendono l’architettura leggibile e pulita, ed anche il metodo getScore() può essere invocato in ogni momento dal controller e questo, secondo noi, è molto utile ed elegante.