Peer-Review 1: UML

Mattia Campana, Matteo Bettiati, Gionata Brebbia, Alessio Caggiano

Gruppo AM01

Valutazione del diagramma UML delle classi del gruppo AM10.

# Lati positivi

Da quello che abbiamo potuto osservare abbiamo ricevuto un UML molto completo sotto svariati punti di vista.

Nel model sono stati presi in considerazione diversi casi limite e/o fasi della partita non banali. Lo si denota dalla presenza delle tante eccezioni, di attributi di stato e metodi di supporto. Inoltre, ci sembra chiaro che abbiano già pensato alle funzioni aggiuntive da voler implementare strutturando in progetto sulla base di quest’ultime.

Nel dettaglio l’uso del UUID l’abbiamo valutato efficace per avere un riferimento univoco per ogni oggetto. Anche la classe Score per differenziare la provenienza dei punti ottenuti dal singolo giocatore.

Per quanto riguarda il controller, anch’esso denso di classi e settorizzato, abbiamo apprezzato l’organizzazione dei diversi metodi e fasi di gioco attraverso una separazione delle classi chiara e cooerente.

# Lati negativi

Tra le critiche costruttive che possiamo rivolgere c’è da far notare l’inconsistenza del design, talvolta con classi di gioco eccessivamente specializzate in confronto ad altre.

Per esempio, riteniamo sia doverosa l’implementazione della classe Bag, possedendo metodi e attributi propri (in questo UML relegata ad una lista nella classe Board). In contrasto con la classe Score, a nostro avviso decisamente utile, ma non necessaria.

Per quanto riguarda il controller ci risulta assente l’astrazione necessaria per poter implementare sia Socket che RMI all’interno della stessa partita (da specificare che l’UML ricevuto fa testo alla versione approvata durante il laboratorio del 21/03, ben prima dell’esercitazione sulla connessione, e quindi non possiamo sapere come il gruppo AM10 abbia modificato in seguito la propria architettura).

# Confronto tra le architetture

I due UML presentano molti punti simili eccetto alcune differenze. Tra queste alcune di quelle che riteniamo valide sono:

1. Classe Score: a cui non avevamo pensato, permette di differenziare i diversi tipi di punti ottenibili da un Player. La riteniamo funzionale soprattutto per la futura implementazione della View (versione GUI).
2. UUID: l’attributo UUID permette di avere un riferimento univoco ad ogni oggetto. Utile nel nostro caso per la persistenza alle disconnessioni, che permette ai giocatori di ricaricare la partita che più preferiscono (considerando il caso in cui gli stessi player abbiano iniziato più partite e poi abbandonate).
3. Gestione connessioni: troviamo interessante la gestione delle connessioni all’interno di GameHandler. Non siamo sicuri di volverle implementare come i colleghi ma di sicuro li terremo come punto di riferimento.