Peer-Review 1: UML

Guido Baldessari, Lorenzo Benzoni, Giacomo Cartechini

Gruppo AM4

Valutazione del diagramma UML delle classi del gruppo AM34.

Lati positivi

- Forte modularità del design
- Buona la scelta di usare l'ereditarietà per differenziare tra modalità semplice ed esperto nelle varie classi.
- Uso di interfacce e dell'ereditarietà rende pulito lo schema.

Lati negativi

- Enumeratore NUMPLAYERS poco utile, basterebbe usare degli integer.
- Consigliamo di implementare *OwnerOfTowers* e *PawnOwner* come interfacce e di fare in modo che *Plance* (o altre classi come *StudentStack*) implementi entrambe queste interfacce.
- Utilizzo dell'inglese non corretto (la parola "Plance" non esiste), consigliamo anche di cambiare *OwnerOfTowers* in *TowersOwner* per uniformità con *PawnOwner* e anche per comodità nella programmazione.
- Classe Board playerListActionPhase: List accetta una sola classe come Generic, non una tupla.
- Classe Board -> utilizzare o un costruttore, o una factory per inizializzare l'oggetto, non utilizzare metodi public "*initialize*".
- Anziché creare molte classi del tipo *EffettoXX*, implementare una singola classe che prenda in ingresso una funzione lambda per generalizzare gli effetti delle diverse carte della modalità esperto.
- Sono presenti alcuni metodi che non ha senso esporre all'esterno, tipo *setCharacterDeck()*, dato che la sua funzionalità può essere svolta dal costruttore al momento dell'inizializzazione della classe di appartenenza. Inoltre, ciò non è "safe", dato che non c'è un motivo lecito per invocare tale funzione nel corso della partita.
- Consigliamo di utilizzare ereditarietà per la classe *Island* (aggiungendo una classe *expertIsland*), in particolare per la presenza dell'attributo *banCard* anche nella modalità semplice.
- Data la presenza di *IslandManager*, consigliamo di far gestire a questa classe tutta la complessità data dalla vicinanza delle isole e dall'unione delle stesse. Sconsiglieremmo i riferimenti all'interno della classe *Island* come *pastIsland* e *nextIsland*, dato che la loro funzione potrebbe essere svolta da *IslandManager.*
- Troviamo che le interfacce *TurnState* e *Action* e le classi che le implementano non abbiano senso, in particolare lo svolgimento del turno diventerebbe molto macchinoso con la creazione di un oggetto per ogni diversa fase di gioco e ogni diversa azione.

Confronto tra le architetture

- Abbiamo trovato che rispetto al nostro design, è utile la presenza dell'enumeratore *Magicians.*