Peer-Review 1: UML

Luca Petrucci, Simone Ponginibbio, Matteo Passoni, Paolo Danilo Secci

Gruppo 65

4 Aprile 2022
Valutazione del diagramma UML delle classi del gruppo 10.

Lati positivi

- **Characters**
  Gestione della problematica relativa alle carte personaggio tramite il pattern Factory.

- **GameParameters**
  Progettazione delle classi GameParameters e ExpertGameParameters per la gestione sia dei parametri della partita, sia delle abilità delle carte personaggio.

- **GameController**
  La gestione del Model è spartita tra due classi, GameController e TableController, ciascuna delle quali ha un ruolo ben definito e distinto da quello dell'altra.

- **GamePhase**
  L'enumerazione delle fasi di gioco permette un controllo più chiaro e semplice della partita.

- **SchoolBoard**
  Progettazione della classe SchoolBoard con conseguente alleggerimento della classe Player. A nostro avviso, questa scelta progettuale opera un'equa ripartizione dei compiti tra le due classi, evitando di incorrere in una situazione di sovradimensionamento di classe.

Lati negativi

- **DiningRoom**
  La classe DiningRoom non è necessaria, avendo un solo attributo, e potrebbe essere totalmente inglobata dalla classe SchoolBoard. Per lo stesso motivo, la classe Bag potrebbe essere integrata nella classe TableController.

- **Modalità Esperto**
  Le sottoclassi ExpertPlayer, ExpertGameController e ExpertTableController potrebbero essere facilmente rimpiazzate da un singolo attributo di tipo boolean IsExpert, da aggiungere alla classe GameController, che imposta la tipologia di partita (per esperti oppure no).

- **PlayerCountIcon**
  Questa enumerazione non sembra essere utilizzata all'interno del modello.

- **Altro**
  Il metodo tryUnifyIsland ha un parametro di tipo Tower che rappresenta il colore di una torre. Secondo il nostro parere, potrebbe rivelarsi più funzionale un parametro che indichi la posizione dell'isola su cui si è fermata madre natura, rendendo più agevole l'ispezione delle isole adiacenti.

Confronto tra le architetture

La nostra scelta di implementare la partita a 4 giocatori ci ha condotti alla progettazione di una classe Team, che racchiude un insieme di giocatori e le corrispettive torri.

In questo modello non vi sono classi specifiche per le pedine di gioco colorate (studente, professore, torre); esse sono rappresentate unicamente mediante il colore, unico fattore di contraddistinzione.

La gestione delle incongruenze tra i match a 2/4 e 3 giocatori differisce tra i due modelli. Il nostro prevede una sottoclasse di Match che gestisce l'assegnazione del numero iniziale di torri e il popolamento delle nuvole nel caso di partita a 3 giocatori, mentre quello in esame adopera una classe GameParameters, con al suo interno tutti quei parametri che cambiano a seconda del tipo di partita.

Un'altra differenza è la modellazione delle carte personaggio. Il nostro gruppo ha scelto di rappresentare ogni personaggio con una diversa sottoclasse di Character o StudentCharacter (entrambe abstract), che equivalgono rispettivamente a CharacterCard e CharacterWithSetupAction. Il modello in revisione invece adotta il pattern Factory per la loro istanziazione.