Descrizione UML

Il nostro progetto di Codex Naturalis utilizza il pattern Model View Controller, nella versione che vi presentiamo il Model e il Controller sono suddivisi in due packages. Nel Controller sono presenti la classe Initializer che a inizio partita prepara il set-up di gioco, la classe GameController che invece gestisce la dinamica di una singola partita (attributo game) attraverso i suoi metodi che modificano lo stato del model e verificano che la logica sia corretta, e la classe TurnHandler per la gestione dei turni di gioco. Per come abbiamo scelto di implementare il controller esso contiene la parte più corposa del codice, ovvero qui avvengono la maggior parte delle operazioni e controlli lasciando al model solo gli osservatori e i setter.

Presentiamo una descrizione dei metodi più rilevanti di GameController:

- drawCard: prende come parametri il giocatore che pesca la carta e il mazzo da cui vuole pescarla, inserendola nella mano del giocatore, e restituisce la carta che viene pescata
- movePawn: prende come parametri il player e il numero di punti da aggiungere alla sua board, implementato nella grafica
- calculateCommonObjectiveScore calcola il punteggio di un singolo giocatore riguardo agli obiettivi comuni, prende come parametri il player, entrambi gli obiettivi comuni passati come carte e restituisce un int
- calculateSecretObjectiveScore: calcola il punteggio di un singolo giocatore riguardo all'obiettivo segreto scelto in partenza, prende come parametri il player e la carta obiettivo segreto e restituisce un int
- addPoints: calcola i punti da aggiungere dopo aver piazzato una carta, facendo i vari calcoli nel caso in cui sia necessario
- setPointsPlayer: imposta sulla board il punteggio il punteggio del player aggiungendo il punteggio della mossa al punteggio precedente del player
- addCard: piazza la carta che gli viene passata nel Manuscript del giocatore

N.B.: Per una scelta di praticità non abbiamo istanziato getter e setter nell'UML del Model

Nel Model abbiamo istanziato una classe Console che gestisce le partite multiple, le restanti classi, che fanno riferimento a una singola partita, sono organizzate in packages: Game, Card, Enum.

Il primo package Game è quello principale contenente le componenti di maggior peso, prima fra tutte la classe Game, la quale contiene i riferimenti a quasi tutte le classi presenti, per avere tutte le informazioni accessibili riguardo a una partita. La Board è il tabellone segnapunti, Market è la zona di gioco dove ci sono i mazzi e le carte girate che si possono pescare (resource 1, resource 2, gold 1, gold 2). In questa classe gli ArrayList sono più consoni come struttura dati per la gestione delle carte nel market e per i metodi che le prendono come parametri. La classe player contiene i dati del giocatore e ha un riferimento al Manuscript, una matrice di Face, che è il set di carte piazzate dal singolo giocatore e che contiene solo metodi osservatori.

Il Package Enums contiene le varie enumerazioni dei simboli, colori e moltiplicatori, in più GameState raggruppa i vari stati del gioco.

Il Package Card suddivide le Objective Card dalle altre. In generale una carta è composta da due facce, front e back, e dal nostro punto di vista la suddivisione è utile per la realizzazione del Manuscript. Le varie tipologie di carte estendono l'abstract Card, eccetto per GoldCard che estende ResouceCard, vista la somiglianza. Per implementare l'ObjectiveCard abbiamo scelto di usare uno Strategy Patter così da semplificare il processo. L'effettiva istanziazione delle carte viene svolta dalla classe JsonParser.