Prova finale di ingegneria del software

2018

Nicolò Felicioni, PeiQing Gao, Davide Yi Xian Hu

Sommario

Model

Public e private objective card

Le carte pubbliche e private possiedono un metodo per calcolare i punti.
Utilizziamo lo strategy pattern per definire l'algoritmo con il quale questi punti vengono calcolati.

Tool card

Le tool card sono caratterizzate da 'aspetti anagrafici' come nome, descrizione testuali e attributi per indicare il costo della tool card e se è stata usata durante il corso della partita.

Round track

Il round track è un array di liste. Ogni cella dell'array corrisponde a un round del round track, mentre la lista contenuta nella cella rappresenta i dadi inseriti nel round del round track.

Dice bag

Il dice bag rappresenta un'astrazione del sacchetto dei dadi. Non istanzia realmente i 90 dadi, ma inizializza invece una lista di 5 interi (uno per ogni colore), tutti inizializzati a 18 (il numero iniziale di dadi per ogni colore).
Ogni volta che si pesca un dado, la classe genera casualmente un colore e sottrae 1 all'intero corrispondente al colore selezionato. Dopodiché genera un dado del colore selezionato e di un valore casuale.

Window pattern

La window pattern è composta da un nome, una difficoltà e un doppio array di spazi.
All'interno di questa classe offriamo i metodi per controllare se un dado è posizionabile rispettando diverso regole e i metodi per effettuare il posizionamento del dado.
Gli spazi sfruttano lo strategy pattern e si dividono in spazi bianchi (BlankSpace), spazi colorati (ColorSpace) e spazi con restrizione di valore(ValueSpace).

Player state

Utilizziamo il player state, seguendo lo State Pattern, per salvare informazioni relative al player e alle sue possibilità di effettuare le diverse azioni nel gioco come posizionare un dado, usufruire dell'effetto di una tool card e passare il turno.

Un caso particolare di player state è la chooseWindowPatternState che obbliga il player a scegliere una window pattern.

Controller

Il controller è la classe che si occupa di gestire la logica del gioco. In particolare, è strutturato attraverso il pattern Observable - Observer per la gestione degli eventi del gioco inviati dalla view. Possiede un timer, alla cui scadenza notifica la disconnessione del player.

Card factory

La card factory (Factory pattern) è un componente usato dal controller per gestire l'inizializzazione di una partita. La factory simula un mazzo di carte, dunque alla costruzione avrà le carte del gioco che a mano a mano verranno distribuite ai giocatori (carte obiettivo private e window pattern) o inserite nel gioco (carte utensile e carte obiettivo pubblico).

Scheduler

Lo scheduler è un componente di supporto al controller e ha lo scopo di gestire i turni del gioco. Fornisce tutte le informazioni riguardanti i turni e i round. È stato realizzato prendendo come riferimento l'Iterator pattern (possiede metodi come hasNext e next).

Network

Il network che abbiamo implementato gestisce connessioni sia RMI che socket.
La particolare implementazione delle due tecnologie è stata progettata prendendo spunto dallo Strategy pattern, ovvero il funzionamento della rete è trasparente agli utilizzatori in quanto utilizzano l'interfaccia (ClientInterface e SessionInterface).
Gestisce la connessione, la disconnessione e la riconnessione da parte di un client verso il server. Per ogni richiesta di connessione del client, il server crea un'apposita sessione lato server.

Connessione

La connessione del client verso il server avviene attraverso l'utilizzo di un username univoco, se tale identificatore è già stato utilizzato da un altro giocatore, il server rifiuterà la connessione.

Disconnessione

La richiesta di disconnessione può essere fatta sia dalla macchina client che da quella server. Un client si disconnette quando termina la partita o quando (solo da GUI) decide di chiudere l'applicazione. Le chiusure improvvise (CTRL+C o crash del calcolatore) di un'applicazione potrebbero non portare il client ha effettuare la procedura di disconnessione. In questi casi sarà il server che dopo aver atteso una predeterminata quantità di tempo, procederà a disconnettere il giocatore.

Riconnessione

Un client che risulta già presente in una partita e disconnesso, può decidere di riconnettersi alla partita. Se il client non risulta disconnesso il server rifiuterà la riconnessione. La riconnessione può avvenire sia tramite la tecnologia RMI che socket, indipendentemente da come la connessione iniziale è stata effettuata.

RMI

La connessione RMI è strutturata come una serie di Observer-Observable. La view nel client inoltra un evento alla RMIClientInterface che a sua volta lo inoltra al RMIServerSession.
La sessione RMI infine procede a inoltrarlo al controller nel server. La comunicazione da server a client funziona in modo simile.

Socket

La connessione socket prevede lo scambio di messaggi in formato json tra server e client. Per trasformare un evento nel formato json testuale, e viceversa, utilizziamo la libreria GSon e quando tale libreria non è in grado di tradurre correttamente l'oggetto adottiamo diversi adapter.

Server

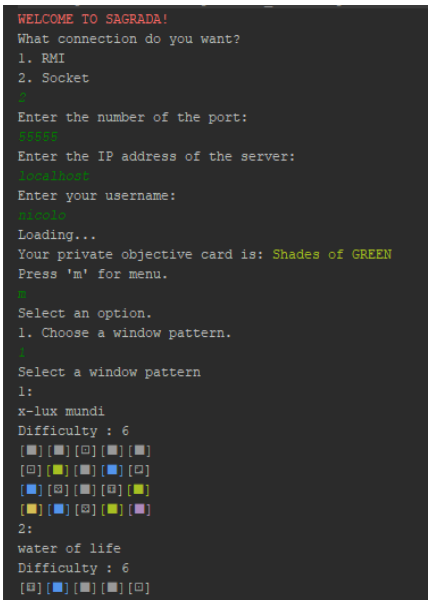
- Viene istanziato una sola volta.
- È in grado di gestire più partite alla volta.
- Supporta sia RMI che socket, e una tipologia di connessione può sostituire l'altra a partita in corso (se vengono soddisfatti i requisiti di riconnessione).

View

La view è il componente alla quale vengono notificate le modifiche del Model e gestisce gli input dell'utente. L'implementazione della view è nascosta agli altri componenti del MVC (+ network) in quanto essi utilizzano solo i metodi offerti dall'interfaccia.

CLI

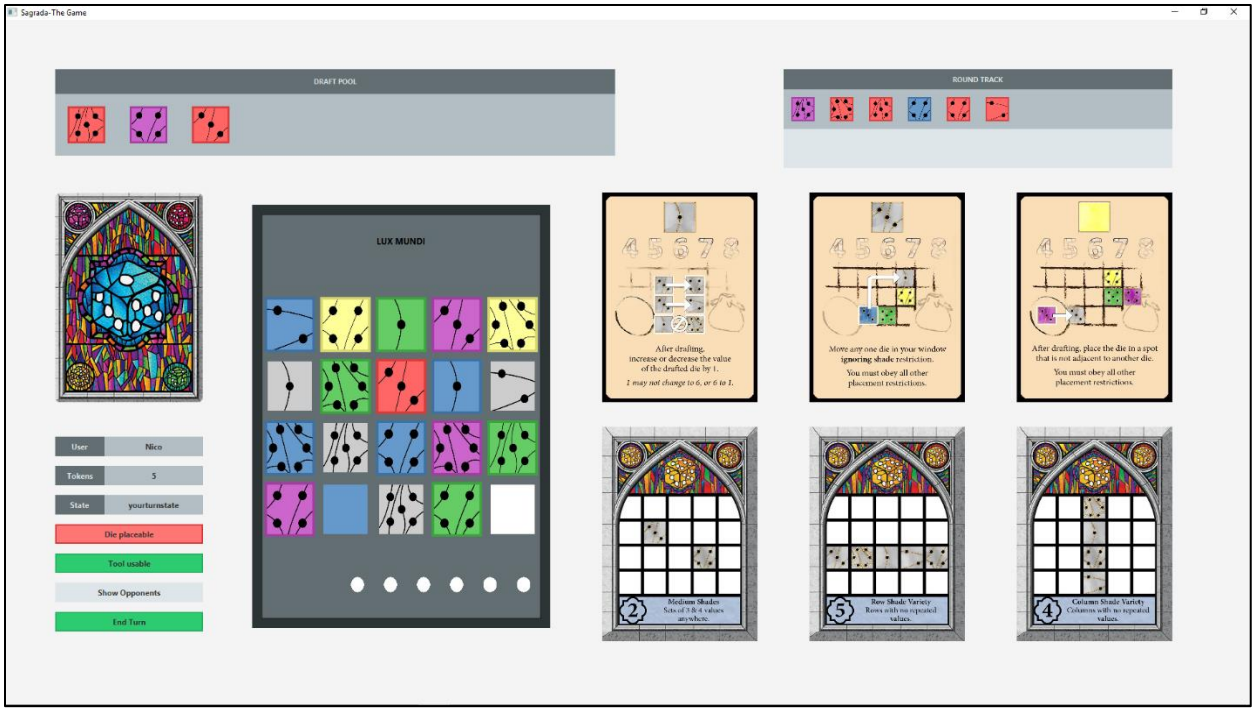
La command line interface è stata realizzata usando la libreria esterna JAnsi, utilizzata per le stampe a colori e i caratteri UTF-8 per realizzare i dadi.
Fa uso di una classe di supporto per le stampe chiamata Printer.
Contiene una classe PlayerMenu che genera dinamicamente le opzioni disponibili (in base alle informazioni contenute nel model) alla richiesta dell'utente.



GUI

La graphic user interface è stata realizzata con l'aiuto di JavaFX.

- Abbiamo realizzato una versione grafica per la maggior parte dei componenti del Model. Tali componenti grafici vengono usati più e più volte in diverse maschere della GUI.
- Diversi componenti grafici sono generati in maniera dinamica (draft pool, round track, window pattern) mentre altri sono semplicemente una immagine (tool cards, objective card).
- Le maschere per utilizzare gli effetti delle tool card hanno un controller che cerca di aiutare l'utente. Ad esempio, selezionando un dado, vengono illuminati gli spazi dove poter spostare questo dado (GUI-MoveDie).



La GUI possiede tre interfacce utenti principali:

- Interfaccia login. Attraverso questa maschera l'utente inserisce nome utente, e gestisce la connessione al server.
- Interfaccia di scelta della window pattern. Tramite questa maschera, l'utente effettua la scelta della window pattern con cui giocherà.
- Interfaccia principale del gioco. È l'interfaccia principale con il quale l'utente interagisce per svolgere le azioni principali della partita.

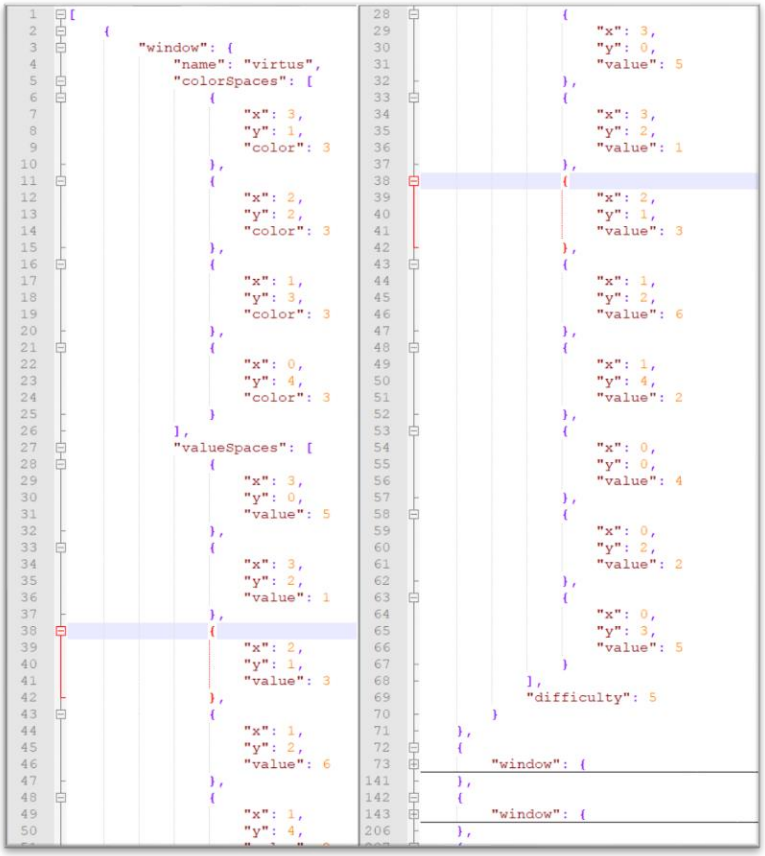
Funzionalità avanzate

Carte schema dinamiche

Abbiamo implementato la possibilità di poter caricare delle window patterns personalizzate.

La view (sia la cli che la gui) disegna tali pattern dinamicamente.

All'interno del file json vengono descritti il nome della window pattern, gli spazi con restrizioni di colore e valore, e la difficoltà della carta.



Partite multiple

La game room è la struttura che abbiamo realizzato per gestire più partite in contemporanea.

- Se esiste una game room in fase di avvio, e un giocatore si connette, tale giocatore viene aggiunto alla game room.
- Se i giocatori sono almeno 2, la game room fa partire un timer che quando scade, avvia la partita.
- Se i giocatori nella game room sono 4, la partita comincia automaticamente.

Avvio del JAR

Sono presenti due jar, uno per il server e uno per il client, e ciascuno possiede un file di configurazione per essere avviato.
Il file di configurazione del server permette di impostare le porte che i server RMI e Socket utilizzano, i timer del gioco (della game room e del turno di una partita) e una variabile booleana per indicare se utilizzare il file di window pattern personalizzate.
Il file di configurazione del client permette di decidere se avviare un client nel terminale o come applicazione grafica. *(I file di configurazione sono situati sotto la directory /src/main/resources/)