HORSE FEVER

Relazione progetto prova finale

Implementazione

Nella realizzazione del progetto abbiamo implementato il modello MVC classico. La vista comunica esclusivamente con il controller e il controller con il modello logico.
La separazione dei tre elementi del pattern MVC è marcata, e ogni flusso informativo tra i livelli vista e modello passa attraverso il controllore, il quale non viene mai bypassato.
In questo modo è garantita la completa consistenza del modello logico e la centralità del controllo. Siamo consapevoli, d'altra parte, che la politica adottata può comportare una diminuzione delle performance.

Vista

La vista è stata implementata con le librerie swing senza uso di software di supporto. La vista possiede il riferimento al controller. E' stata creata un'interfaccia per consentire la realizzazione di altre viste che funzionino con il nostro controller.

Controller

Il controller nel nostro progetto ha il compito di gestire il flusso del gioco, facendo riferimento ad uno stato, e di fare da tramite tra la vista e il modello logico, garantendo così la consistenza delle modifiche apportate al modello logico.
Nel controller, il flusso dell'esecuzione segue una sorta di ciclo tra i metodi interni, in particolare il metodo continueGame() si occupa di leggere lo stato e in base ad esso decidere cosa fare. continueGame() è chiamato da molti metodi che, dopo aver svolto la propria attività ed eventualmente cambiato lo stato, delegano a continueGame() il compito di "andare avanti". Questo flusso costituisce un ciclo, voluto, per mantenere in un unico metodo il core del controllo, al fine di garantire la scalabilità e in generale la facilità di apportare modifiche al flusso stesso.
Abbiamo creato un'interfaccia che consente la realizzazione di altri controller che possano comunicare con la vista.

Modello

Il modello logico si occupa di mantenere le strutture dati nelle quali sono contenute tutte le informazioni sullo stato di tutte le componenti del gioco. Il modello logico è stato strutturato gerarchicamente: esiste una classe, *Game,* che contiene riferimenti a tutte le altri classi. Ogni funzionalità è stata implementata al livello logico adeguato.
Es: Gli effetti delle carte azione coinvolgono a volte la singola corsia, a volte il contesto di tutte le corsie, altre volte agiscono sullo stato del gioco cambiando, ad esempio, le quotazioni.
La suddivisione di queste funzionalità per livelli logici consiste nell'implementare l'effetto di queste carte direttamente nella classe coinvolta dalla specifica azione.

Regole di visibilità

La politica adottata è quella dell'information Hiding. Nel codice gli attributi delle classi logiche sono sempre stati dichiarati *private*. Abbiamo utilizzato il metodo clone() (spesso opportunamente sovrascritto) quando si rendeva necessario il passaggio di valori che non dovevano essere modificati.

RMI

Per la realizzazione del paradigma RMI ci siamo trovati di fronte ad un trade-off tra buona organizzazione del codice e sua duplicazione.
Abbiamo ritenuto necessario implementare nuove interfacce (per la vista e per il controllore) per gestire la chiamata remota dei metodi attraverso il paradigma RMI: sia perchè le interfacce richiedono l'estensione della classe *Remote* per gestire i metodi da remoto, sia perchè la gestione remota del gioco richiede la modifica di alcuni prototipi dei metodi contenuti nell'interfaccia implementata nella versione locale. L'utilizzo di nuove interfacce è anche dovuto al fatto che ogni metodo che si intende chiamare da remoto deve lanciare *RemoteException*: sarebbe stato inutile modificare le interfacce del gioco locale inserendo il lancio di questa eccezione nei vari metodi quando nel caso di gioco locale non verrebbe mai lanciata.
Questa scelta ha tuttavia portato all'aumento del numero di cicli tra pacchetti (senza la parte RMI ci sarebbe un unico ciclo tra controller e vista) e alla duplicazione di parte del codice delle classi che gestiscono il controllore e la vista, rimane invece unica la parte logica alla quale i due tipi di controllore fanno riferimento.