Relazione progetto JTrash

Tommaso Mattei - 1884019 - Corso: Presenza MZ

Febbraio 2024

Contents

1 Decisioni progettuali

1.1 Gestione del profilo utente

La gestione del profilo utente è stata affrontata principalmente nelle classi **Utente**, **ProfileView**, **ProfileController** e **ChooseNicknameView**.

Come si può intuire dal nome delle stesse è stato applicato il principio di MVC come verrà mostrato in seguito; infatti Utente corrisponde al modello del design pattern.

Alla creazione di utente come prima cosa si fa riferimtno ad un file di testo presente nel progetto per controllare se, dato un id utente, sono già presenti delle informazioni da recuperare nel sistema. Nel caso affermativo si salvano le informazioni già presenti in una lista apposital, le quali potranno essere ottenute tramite vari getter nel controller per la view. In caso contrario invece si scriverà sul file di testo inizializzando le informazioni in modo basilare: il livello sarà posto ad 1, le partite vinte e perse saranno 0, mentre l'immagine del profilo sarà impostata con un'immagine standard. In merito a ciò il modo sfruttato per salvare quest'ultima informazione è tramite una stringa la quale tiene il path assoluto (se data dall'utente) così da poterla poi visualizzare.

Le informazioni presenti nella classe Utente e poi nel file di testo vengono aggiornate in base al risultato della partita che si può giocare (a cui l'utente è ovviamente collegato) ma anche grazie al controller che tramite la view accetta degli input relativi al nickname e all'immagine profilo.

All'interno della view sono quindi presenti due pulsanti per adempiere alle funzioni potendo rispettivamente accettare una stringa per il nickname o potendo far scegliere dall'utente un file adeguato. Nel caso del file tutto è eseguito con soltanto la view, il controller ed il modello mentre invece per il nickname ci si serve di un frame aggiuntivo che appare richiedendo l'informaizone necessaria; esso è ChooseNicknameView.

Tutte le informazioni riportate nella view sono caricate tramite il controller tranne il numero di partite totali che può facilmente essere inferto conoscendo gli altri dati.

Si parlerà successivamente più in generale della GUI la quale è abbastanza costante nei vari frame presentati per interagire con l'utente.

1.2 Gestione di una partita completa

La gestione dell'intera partita di Trash è il fulcro dell'intero progetto e pervade tutte le componenti dello stesso, per questa sezione in particolare però si parlerà concentrandocisi quasi unicamente sull'aspetto del modello sottostante a tutto; in particolar modo si parlerà delle seguenti classi:

- **Carta**
- **CPU**
- **Game**
- **GenericPlayer**
- **Giocatore**
- **Mazzo**
- **SemeCarta**
- **Utente**
- **ValoreCarta**

Partendo dalle componenti più basilari su cui si base il resto dell'applicazione possiamo vedere Carta, ValoreCarta e SemeCarta insieme a Mazzo.

Senza dilungarsi più del dovuto nelle due enumerazioni abbiamo la lista dei semi delle carte e dei valori delle carte rispettivamente; per quest'ultime è presente sia una conversione in stringa che in intero, un dettaglio che renderà più semplice implementare la classe Game ma anche il caricamento delle immagini presenti tra gli asset.

La classe carta unisce in sé le due informazioni precedenti e a sua volta è elemento fondamentale della classe mazzo che presenta numerosi metodi importanti come mischiare, decidere la grandezza del mazzo, riempire le mani dei giocatori e simili.

Ora che si posseggono le fondamenta della classe Game è necessaria una triade di classi per poter effettivamente lavorare sulla logica di gioco: le classi dei giocatori. Per ordine mentale, anche se forse non strettamente necessario, si è creata una classe da giocatore generico che presenta i metodi e le informazioni in comune tra una CPU e un giocatore vero e proprio comandato dall'utente che sta usufruendo dell'applicazione. Come appena menzionato in quanto a informazioni l'unica differenza è il collegamento ad un utente o meno, oltre a ciò però sono i metodi e l'implementazione degli stessi che divergono totalmente per fronteggiare l'input umano.

I due metodi più importanti del gioco sono presenti all'interno del giocatore: un metodo per scartare e un metodo per eseguire l'azione base nello giocare a trash: prendere la carta che si ha in mano e selezionare la posizione adatta dove inserirla, per così prendere la carta coperta sul tavolo e metterla in mano. Qui sono presenti varie eccezioni personalizzate per rispettare le regole del gioco ed

oltre a ciò si trova qui la logica nel gestire le varie carte da gioco che precedentemente abbiamo creato. Ai fini del trash il seme della carta non è davvero importante quindi qui si gestiscono i vari valori che la carta può assumere così come la possibilità o meno di essere inserita in una certa posizione.

Pensando un momento alla logica generale del gioco, è qui che viene chiamata una routine adatta a controllare se si è fatto trash o meno, il quale setterà varie flag che inseriranno il gioco nella modalità speciale dell'ultimo turno prima dell'inizio di una nuova manche.

E' bene evidenziare qui la differenza tra il metodo per mettere la carta della CPU e del giocatore, così come il metodo di scarto: il giocatore ha la libertà di dare vari input per la carta potendo sbagliare mentre per la cpu ogni cosa è eseguita in automatico tornando però un booleano per comprendere l'esito positivo o meno del metodo, un qualcosa che sarà utile pensando alla gestione dei turni della cpu.

Ora che ogni cosa è al suo posto si può parlare di più sulla gestione della partita in toto. I metodi a disposizione del giocatore quindi sono soltanto tre: pescare, scartare e posizionare una carta nel giusto punto. Prima di parlare della gestione dei turni della cpu ed il raggiungimento del trash si ricorda come all'utente è concesso all'inizio della partita scegliere il numero di giocatori totali fino ad un massimo di 3 avversari (quindi 4 giocatori totali); in base a ciò il numero di carte nel mazzo così come il numero di avversari cambia.

Pensando alla gestione dei turni della CPU sfruttiamo l'azione che determina la fine del turno del giocatore: lo scarto. Alla fine dello scarto del giocatore la routine della CPU viene invocata e così facendo essa scorre la lista di giocatori fino a quando non incontrerà non una CPU ma un Giocatore, fermandosi. Il turno di una singola cpu è un loop alimentato finché è possibile inserire carte nella giusta posizione, fermandosi solo privato di questa possibilità, scartando e passando la mano alla prossima CPU.

Si esamina ora l'idea dietro il Trash e dell'ultimo round concesso dopo che un giocatore lo ha eseguito. Quanbdo qualcuno esegue un trash si memorizza il turno di chi lo ha fatto; all'inizio di ogni turno in generale si controlla sempre se ci si trova nello stato di trash e se sì si quando si raggiunge il turno iniziale si considera finita la manche, chiamando l'apposita routine. Lo stato di trash è fondamentale per far sì che, se anche qualcun altro dovesse farlo a sua volta il turno continuerà naturalmente fino alla sua fine senza ulteriori avvisi.

In questo modo raggiungiamo la routine che pone fine alla manche e dà inizio alla prossima. Ogni cosa è resettato al suo punto di partenza se non per una differenza: i vincitori della manche precedente vedono eliminato un posto dalla propria lista mano. Grazie a questo fatto è rivelata la fine del gioco: ogni volta che si fa trash esiste un controllo che osserva la lunghezza della mano del giocatore...se essa è pari a 1 allora il gioco è finito e si conclude con un'ultima routine che interessa particolarmente all'utente.

Giunti fin qui si controlla se il vincitore è una CPU o un giocatore. Nel caso della CPU una sconfitta viene aggiunta all'utente mentre nel caso di vittoria si fa lo stesso, aumentando però anche il livello in base al numero di avversari sconfitti.

1.3 Adozione di JavaSwing per la GUI

Classi rilevanti:

- **ChooseNicknameView**
- **FinalFrame**
- **GameView**
- **IntermediateWindow**
- **MainMenuView**
- **ProfileView**
- **StartUpIDFrame**

Fino ad ora si è parlato solamente del lato relativo al modello sottostante accennando la GUI soltanto nella parte relativa al profilo dunque è tempo di approfondire.

La GUI è stata creata grazie a JavaSwing, presentando più o meno sempre lo stesso stile attraverso i vari frame mostrati a schermo con una finestra 1024x768 la quale non può essere ridimensionata ed uno sfondo grigio per ogni cosa. Attraverso il progetto si è fatto uso principalmente di un layout manager nullo per una maggiore rapidità e libertà impostando direttamente i valori assoluti dei vari label e bottoni presenti. Ovviamente questo porta lo svantaggio relativo al ridimensionamento della finestra la quale è bloccata onde evitare stravolgimenti.

Il passaggio da una finestra all'altra è sempre effettuato grazie ai bottoni e agli ActionListeners posti su essi i quali non solo passano le rilevanti informazioni tra le varie schermate ma settano la visibilità on e off in base all'input. E' proprio grazie a questo passaggio di informazioni che all'inizio l'utente è in grado di inserire il proprio id per fare accesso e ritrovarsi nella schermata principale dell'applicazione dove potrà o accedere alle informazioni del profilo o alla creazione di una nuova partita. Creando una nuova partita sarà presente una piccola finestra che permetterà l'inserimento del numero di giocatori totali presenti nel game, dando inizio ad ogni cosa.

Attraverso la MenuBar nell'intera GUI è perennemente presente una barra con scritto menù da cui accedere a due semplici ma fondamentali bottoni che possono o chiudere l'applicazione o possono riportare in qualsiasi momento alla home.

Dal punto di vista della densità di componenti la classe più importante della GUI oltre a ProfileView è GameView dove viene visualizzato tutto ciò che avviene all'interno del gioco sottostante. Non c'è molto da dire in merito all'implementazione grafica con i vari bottoni che richiamano i metodi più importanti della classe Game e i numerosi label che ospiteranno le carte delle CPU. Importante far notare come le carte sul terreno del giocatore sono tutte quante create come dei bottoni su cui sopra è messa l'immagine della carta.

Un ultima parte della GUI non ancora menzionata è il FinalFrame che ospita il messaggio finale il quale decreta la fine della partita, riportando una vittoria o una sconfitta, dopo la quale l'unica cosa da fare è tornare alla home o chiudere l'applicazione; riparleremo di questo frame nella sezione dell'animazione.

1.4 Riproduzione di audio sample

La riproduzuzione degli audio sample coinvolge principalmente la classe **AudioManager** ma perme completamente l'intera GUI anche se non precedentemente menzionato. Istanziando l'audio manager nei rispettivi frame si è riusciti al premere di ogni bottone di assegnarci un particolare suono e soprattutto grazie alla gestione dell'audio è possibile far partire la musica una volta entrati nel game vero e proprio.

Oltre alla musica di background i suoni sono distinti tra la parte fuori dal game e quella dentro al game dove il suono di carte che si muovono viene sentito dopo ogni interazione del giocatore.

1.5 Animazioni ed effetti speciali

Per via di limitazioni intrinseche alla mia conoscenza in materia non si è riuscito a produrre degli effetti speciali che permeano la game class. In compenso chi riuscirà a raggiungere la fine di una lunga ed estenuante partita a Trash sarà ricompensato osservando nel **FinalFrame** l'implementazione di **AnimationPanel**, un pannello che domina la schermata e mostra una simpatica animazione di un joker che rimbalza per i confini del frame, essendo l'unica carta rimasta fuori dal gioco.

Per quanto riguarda l'implementazione l'animationPanel si basa sulla presenza del metodo Paint e soprattutto di un timer collegato ad un ActionListener. Ogni tot millisecondi l'actionListener è chiamato aggiornando la posizione spaziale dell'immagine e assicurandosi che non esca mai dal perimetro del pannello assieme a repaint. Per ottenere però una costante animazione si sfrutta alla base paint che seppur non venendo istanziato, essendo un panel un component sottostante, riesce ad essere invocato.

2 Design Pattern

Classi rilevanti:

- **Subject**
- **Observer**
- **GameController**
- **ProfileController**

Avendo accennato solo brevemente all'inizio dei controller è tempo di parlare dei Design Pattern utilizzati, così come della utilità assoluta nel permettere l'interazione tra la logica di gioco sottostante e la GUI.

Ogni talvolta si parla di collegare le due componenti principali del progetto si parla sempre del Model,View,Controller pattern, MVC in breve. Il pattern si basa sul modellare separatamente modello e GUI senza che vi sia alcuna interazione tra di loro, ottenendo un'indipendenza che permetterà magari diverse implementazioni future a partire dallo stesso modello.

Il modo secondo il quale le due componenti comunicano fra di loro è il controller. All'interno del progetto questo tipo di design pattern è applicato due volte: rispettivamente nella sezione relativa al profilo e nella sezione del gioco vero e proprio.

Alla creazione degli stessi controller vengono passati sia model che view per creare la citata connessione, ma per far sì che il collegamento avvenga appieno è fondamentale che vi siano degli action listener dalla parte della GUI per captare gli input che successivamente verranno passati al modello. Se da un lato quindi si ottiene informazioni dall'utente passando per la GUI essa stessa non possiede niente in sé e non tiene traccia di nulla essendo solo una tela che verrà dipinta dal controller. Per fare ciò il controller avrà bisogno di essere chiamato in causa dal modello nei momenti opportuni e aggiornare ciò che l'utente vede.

Il modo più consono per effettuare quanto previamente detto è tramite un ennesimo design pattern, il quale è intrinsecamente collegato a MVC: l'Observer pattern. Nell'Observer pattern abbiamo due tipologie di classi (le quali infatti nel progetto sono interfacce che vengono implementate) Observer e Observable. Le classi che implementano Observable sono osservate da X observers, nel nostro caso soltanto uno. Creando un collegamento tra le due, le quali per noi sono rispettivamente controller e model, esiste un metodo update che aggiornerà il controller in alcuni specifici momenti dell'esecuzione del modello, portando al famoso aggiornamento della GUI.

Tramite questi design pattern si ottiene quindi la perfetta armonia e collaborazione tra le varie parti prese in causa.

L'ultimo design pattern di cui bisogna parlare per il progetto è il Singleton pattern. L'unica classe che implementa il Singleton pattern è l'audio manager nel quale è possibile vederne la particolarità: il costruttore è privato, dunque l'unico modo per istanziarlo è tramite il suo metodo getInstance, il quale si assicura che se non esiste venga creato o che se esiste si faccia riferimento a quello già esistente: non possono esistere due istante dell'oggetto allo stesso tempo.

Seppur logicamente questo ragionamento sarebbe potuto essere usato per l'istanziazione della classe Giocatore o della Game class stessa, alla fine si è deciso non intraprendere quella strada poiché dal punto di vista dell'utente era già al di fuori delle sue possiblità ottenere contemporaneamente multiple istanze con cui interagire delle classi.

3 Stream

Classi rilevanti:

- **GenericPlayer**
- **GameController**

Infine è tempo di parlare degli stream, un meccanismo molto utile introdotto in java8 che permette di eseguire numerose operazioni in una sola riga su varie collections.

All'interno del progetto non ne è stato fatto un estensivo uso poiché le varie situazioni incontrate spesso non li vedevano come soluzione predonimante. Comprenndendo ciò e non volendo accanirsi nel trasformare inutilmente codice già scritto in stream per puro gusto essi sono stati sfruttati solo quando sono apparsi come soluzione immediata o addirittura fondamentale.

All'interno del GenericPlayer è apparso come soluzione più ovvia l'utilizzo negli stream per quanto riguarda l'ottenimento di una nuova sottolista mano a partire dalla precedente, eliminando l'ultima posizione.

Invece nell'unica altra istanza di stream si ha una soluzione fondamentale tramite una mappa. L'utilizzo degli stream ha facilitato enormemente l'associazione ad ogni carta di gioco del suo corrispettivo nome in termini "comprensibili" dai file png presenti negli asset. Passando la mano del giocatore e associando ad ogni carta il nome del file si riesce elegantemente poi a settare ogni immagine correttamente sulla GUI.

4 Note progettuali e di sviluppo

Reputo infine di essermi espresso estensivamente sull'intero progetto in merito alle idee in corso d'opera e all'approccio sistematico verso lo stesso, senza dover qui sentirmi di aggiungere altro.