RELAZIONE PROGETTO JUNO

Metodologie di Programmazione 2021/2022 – Canale M-Z

1934897 – Peto Birlea Roxana Sorina

All'interno della cartella src sono presenti un file "data.txt" con i dati di 3 giocatori (utilizzati per i test finali del programma) e 6 sottocartelle, di cui 5 contenenti codice sorgente e risorse per il progetto e una per la documentazione.

Come richiesto nelle specifiche, ho progettato e sviluppato una versione giocabile del gioco UNO in Java, gestendo una partita completa in modalità classica, con un giocatore umano contro tre giocatori artificiali.

DESCRIZIONE DEL GIOCO

- La cartella main contiene la classe JUno, che a sua volta contiene l'unico metodo main del progetto;
- Fatto partire il programma, si apre la schermata iniziale del gioco, contenente soltanto l'immagine del gioco e il pulsante che permette di passare al menu del gioco;
- Una volta aperto il menu le opzioni sono 4: aggiungere un nuovo profilo, selezionare un profilo preesistente, giocare, salvare i dati ed uscire.
- Se si sceglie di aggiungere un nuovo profilo verrà aperta una nuova finestra dove inserire il nickname e scegliere uno tra i sei avatar proposti; è necessario compilare entrambi i campi perché altrimenti non è possibile inserire il profilo nel database e verrà visualizzato un avvertimento;
- Se si vuole selezionare un profilo preesistente verrà aperta una nuova finestra dove poter scegliere il profilo oppure rimuovere uno dei profili esistenti; una volta selezionato il profilo con cui giocare la finestra si chiuderà;
- A questo punto, e non prima perché altrimenti l'utente verrà avvisato che non è stato scelto un profilo con cui giocare, si può proseguire iniziando una partita;
- Durante la partita, per proseguire nel gioco, è necessario premere sul pulsante CONTINUA che si trova esattamente sotto al mazzo di scarto: nel caso in cui sia il turno di uno dei giocatori artificiali il gioco prosegue e il giocatore umano viene notificato delle azioni non visibili dei giocatori artificiali (ad esempio la pesca di una carta), mentre nel caso in cui sia il turno del giocatore umano, se questo ha davanti una carta azione deve premere su CONTINUA per proseguire, altrimenti può scegliere cosa fare (pescare o giocare una propria carta). Durante la partita vengono visualizzati messaggi che informano il giocatore umano delle azioni degli avversari in risposta alle carte azione (ha

pescato due carte, ha invertito la direzione di gioco, salta il turno, ecc.…). Le regole adottate per il gioco sono quelle del manuale ufficiale del gioco, a meno delle due regole interattive sulla contestazione della carta Jolly Pesca Quattro e sulla pesca delle due carte a fine turno quando non è stato premuto il pulsante UNO (vengono sempre pescate due carte ma solo quando il giocatore ha finito le carte).

- Alla fine della partita/delle partite giocate, per salvare i dati dei nuovi giocatori, le nuove informazioni acquisite durante il gioco e chiudere correttamente il programma è necessario usare il pulsante SALVA E CHIUDI, altrimenti i progressi verranno persi.

DECISIONI DI PROGETTAZIONE

GESTIONE DEI PROFILI: Al momento dell'aggiunta di un nuovo profilo è possibile scegliere un nickname e un avatar tra i sei proposti; gli altri attributi (partite giocate/vinte/perse e livello) vengono impostati ai valori di default (rispettivamente 0,0,0 e 1). Il database con i profili dei giocatori, rappresentato dalla classe PlayersDatabase (model), ha due metodi di caricamento e salvataggio dei dati dei giocatori dal e sul file "data.txt", che vengono sfruttati rispettivamente quando viene aperto il menu di gioco e quando viene premuto il pulsante SALVA ED ESCI. L'aggiunta e rimozione dei profili al/dal database è ottenuta mediante la coordinazione tra i frame AddNewPlayer (view), SelectPlayer (view), Controller e PlayersDatabase (model). Per rendere possibile il salvataggio dei dati dei giocatori, la classe Player implementa l'interfaccia Serializable.

DESIGN PATTERN: I pattern adottati per l'intero progetto sono quattro.

- Un pattern architetturale, **MODEL-VIEW-CONTROLLER**: tutte le classi del programma, esclusa la classe JUno, sono divise nelle tre parti model, view e controller. Fanno parte del package model tutte quelle classi che riguardano i dati, la logica e le regole del gioco, dunque tutte le componenti necessarie alla gestione dei giocatori e del gioco. Il package view contiene tutte quelle parti che rendono possibile l'interazione dell'utente con il programma e mostrano a video le parti del model. Infine, il controller fa da intermediario tra model e view, accogliendo richieste e inoltrandole. Ho adottato il pattern per separare e gestire al meglio le varie parti e i compiti all'interno del codice.
- Un pattern creazionale, **SINGLETON**: è stato adottato per le classi Controller e PlayersDatabase, cioè quelle classi per cui deve esistere una sola istanza in tutto il codice dell'applicazione. L'unica istanza viene dichiarata campo static all'interno della classe, viene nascosto il costruttore e l'unico metodo per ottenere un'istanza di questa classe è utilizzando il metodo getInstance(). Ho adottato il pattern per una questione logica e per la comodità dell'avere una sola istanza accessibile da qualunque punto del codice.
- Un pattern comportamentale, **STRATEGY**: è stato utilizzato in vari punti della parte view per assegnare comportamenti specifici ai vari frame/panel/button del progetto. Tramite metodi definiti appositamente è possibile assegnare al componente un'interfaccia che

definisce le azioni da eseguire in risposta al verificarsi di un certo evento. Ho adottato questo pattern per rendere flessibile il comportamento dei componenti.

- Un secondo pattern comportamentale, **OBSERVER**: è stato utilizzato per le componenti della parte view (PlayerPanel e CardsPanel, ovvero i panel con le carte dei giocatori sia umani sia artificiali) che devono ricevere notifiche sul cambiamento di stato di alcune parti del model (giocatore di turno) e per quelle che devono inviare le notifiche (GamePanel). Ho adottato il pattern per rendere uniforme il comportamento di quelle componenti che devono essere modificate in modo simile e che hanno bisogno delle stesse informazioni per l'aggiornamento.

INTERFACCIA GRAFICA: Per quanto riguarda l'interfaccia grafica, ho adottato la libreria Swing, e di conseguenza anche AWT (Abstract Window Toolkit) essendo Swing una sua estensione, e in particolare ho fatto uso di JFrame, JPanel, JButton, JTable, JLabel, JTextField, JRadioButton, JComboBox e varie interfacce.

USO DEGLI STREAM: Ho utilizzato gli stream in vari punti del codice per la lettura/scrittura di dati da/su file e per esaminare/filtrare/fare il mapping di liste di giocatori, di carte, di Stringhe, ecc.