Progetto di gruppo
Programmazione Avanzata Java e C

Stefano Valloncini
Yuhang Ye
Luigi Amarante

Per il corso del professor

Massimiliano Redolfi

Università degli Studi di Brescia

Ultimo aggiornamento: 21 giugno 2022

Indice

1 Introduzione

Questo documento è la relazione del progetto realizzato per il corso di *Programmazione avanzata Java e C* dell'Università degli Studi di Brescia. L'obiettivo del progetto è stato realizzare in Java il gioco *Uno*, rispettando tutti i requisiti richiesti dal docente del corso.

Prerequisiti del progetto

- essere realizzato in Java
- avere un'architettura Client/Server applicativo
- avere un'interfaccia grafica basata su *Java Swing* o *JavaFX* e rispettare il pattern *MVC*
- utilizzare i thread (ad esempio per la comunicazione oppure per la gestione di calcoli complessi/simulazioni)

In modo particolare, abbiamo usufruito per la maggior parte dell'interfaccia graifica, del framework **Swing**, che verrà descritta approfonditamente successivamente nella sezione 2.3.

1.1 Il gioco di carte

Iniziamo con il vedere le regole generale del gioco. A tutti i giocatori vengono consegnate, casualmente, 8 carte. **Il gioco si svolge a turni.**

In un turno un giocatore è libero di pescare o posizionare sul campo di gioco una carta, la quale deve essere ovviamente valida, cioè rispettare le regole del gioco. Una carta non valida non può essere posizionata sul campo di gioco. Il giocatore che per primo riesce a esaurire tutte le carte che ha nella sua mano vince la partita. L'altro giocatore, perde.

Quando un giocatori raggiunge un numero di carte in suo possesso pari a uno, deve attivare il pulsante ***UNO!***. In caso contrario, se non effettua questa operazione, gli verranno assegnate due carte, e quindi, gli viene impedito di vincere la partita in quel turno, nel caso in cui il giocatore in questione avesse scelto di posizionare la sua ultima carta sul campo da gioco.

Non si prevede di continuare il gioco una volta che uno dei giocatori ha completato il gioco: esiste solo un vincitore, e tutti gli altri giocatori sono perdenti.

1.2 Il mazzo di carte

Un mazzo di carte di *Uno* è composto da 108 carte: carte numeriche, carte *action-card*, carte *wild-card*. Tutte le carte, a parte le carte *wild-card*, sono caratterizzate da un particolare colore (e un tipo, che caratterizza il tipo di carta che rappresentano). Le carte *wild-card* non hanno colori, ma una volta

posizionate richiedono che sia selezionato un nuovo colore a propria scelta. L'intero mazzo di carte è presentato nella figura 1.

- 19 carte di colore Blu, numerate dall'1 al 9 (2 serie) più uno 0
- 19 carte di colore Rosso, numerate dallì1 al 9 (2 serie) più uno 0
- 19 carte di colore Verde, numerate dall'1 al 9 (2 serie) più uno 0
- 19 carte di colore Giallo, numerate dall'1 al 9 (2 serie) più uno 0

Inoltre, come carte *speciali:*

- 8 carte **Pesca Due** dei quattro colori sopracitati (tipo action-card)
- 8 carte **Cambio giro** dei quattro colori sopracitati (tipo action-card)
- 4 carte **Cambio colore** (tipo wild-card)
- 4 carte **Pesca Quattro e Cambio colore** (tipo wild-card)

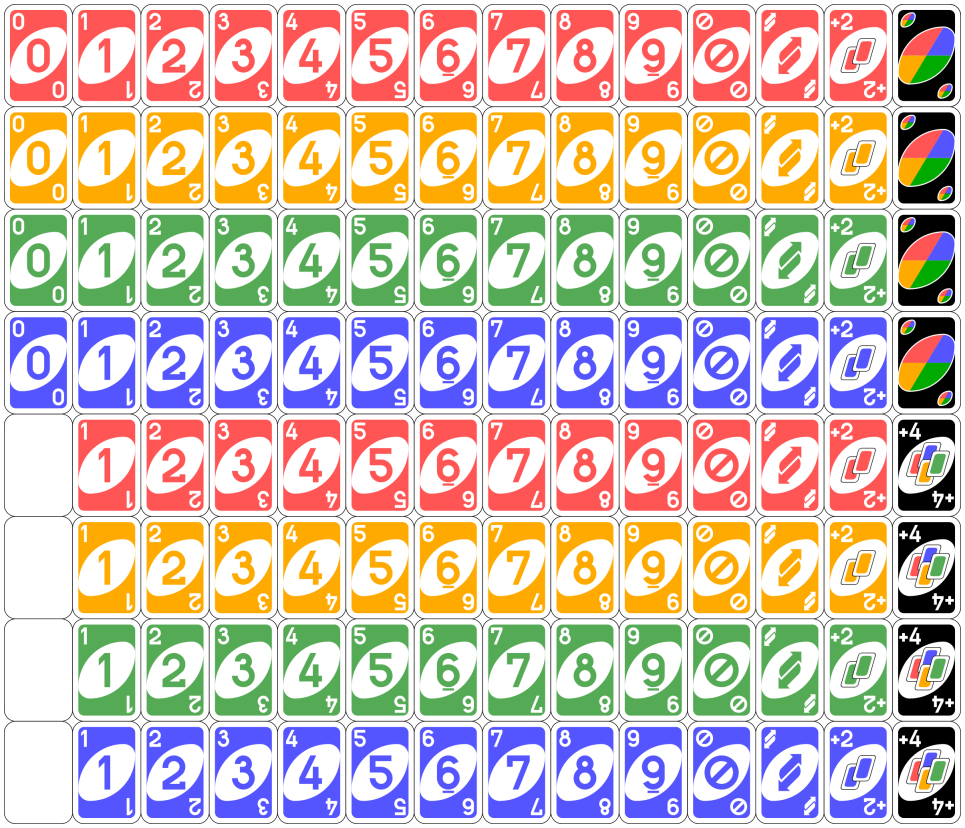

Figura 1: L'intero set di carte di uno

1.3 Successione di gioco

Inizio gioco
Dopo aver consegnato le carte come descritto, inizia un giocatore. L'estrazione del giocatore che inizia per prima è fatta in modo casuale. Viene inoltre estratta dal mazzo una carta in modo casuale, che sarà considerata come la prima carta posizionata sul tavolo.

Turno giocatore
Il giocatore deve decidere in base alle sue carte se pescare una nuova carta dal mazzo, oppure se utilizzare una carta in base a quella presente sul tavolo. Si è limitato il numero di carte a 30 contemporaneamente in mano ad un giocatore, perché la probabilità che non sia disponibile una carta valida tra le 30, è statisticamente impossibile. Questo impedisce ad un giocatore di giocare solo di mosse banali, cioè pescando solo carte.

Prossimo turno
Una volta effettuata una qualunque delle due scelte ammissibili, cioè o pescare una carta o posizionarne una sul tavolo da gioco, il turno passa al giocatore successivo. Unica eccezione è fatta nel momento in cui viene utilizzata dal giocatore una carta "cambio giro" o "stop", in uno di questi due casi il turno viene passato *saltando* il turno successivo per il giocatore prossimo o *invertendo* i turni successivi, come descritto successivamente.

Fine del gioco
Il gioco prosegue finché un giocatore non termina le sue carte nella sua mano. Quando un giocatore arriva ad avere solo una carta nella sua mano, allora deve mobilitarsi per indicare di avere solo una carta a tutti gli altri giocatori. Giocando in persona questo si ottiene esclamando "uno", nel gioco occorre premere il pulsante "Uno!". Il giocatore che con una sola carta compie una qualunque operazione (pescare una carta o selezionarne una), è penalizzato con l'aggiunta di due carte alla sua mano. Un giocatore che indica correttamente "uno", e seleziona la sua ultima carta è vincitore.

1.4 Modalità di gioco

Abbiamo creato una serie di modalità di gioco, in modo da creare diversità nei temi di sviluppo:

- **Multiplayer LAN:** attraverso l'utilizzo di specifiche classi (che verranno descritte in seguito), abbiamo permesso a due utenti di collegarsi e giocare una partita su due diversi computer, collegati sulla stessa rete locale.
- **Multiplayer locale**: abbiamo permesso di far giocare due giocatori sullo stesso computer, che non dispongono di due diversi computer o di una connessione LAN.
- **Singleplayer**: abbiamo implementato una funzione di singleplayer (ovviamente in locale) che permette di giocare contro il computer, o di simulare una partita gestita interamente dal computer. Abbiamo creato dunque una *sottospecie* di intelligenza artificale molto semplice e che

dato lo scenario di gioco fosse in grado di decidere la scelta migliore (o una delle migliori) per il giocatore.

Figura 2: Schermata principale del gioco da cui selezionare la modalità di gioco

2 Pattern Model-View-Controller

Abbiamo utilizzato il pattern architetturale MVC, suddividendo il programma logicamente in view, model e controller.

- il **model** fornisce i metodi per accedere ai dati utili all'applicazione;
- la **view** visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
- il **controller** riceve i comandi dell'utente (in genere attraverso la view) e li attua modificando lo stato degli altri due componenti.

Il framework Swing, che abbiamo utilizzato per la realizzazione grafica, tuttavia, non presenta una chiara separazione tra controller e view. Di fatto delle parti di view sono presenti nel controller. La suddivisione tra controller e model invece è marcata e segue esattamente il pattern MVC. Ovviamente, **model e view non comunicano in nessun modo** (non direttamente, ma avendo come tramite il controller!).

2.1 Funzionamento generale

Ogni volta che l'user che sta utilizzando il gioco, compie un'azione (che ricordiamo, può essere soltanto pescare la carta oppure sceglierne una dal proprio mazzo per eventualmente posizionarla sul campo da gioco se valida), la view lancia un evento, ed, il controller, in ascolto per possibili eventi, recepisce che è avvenuta un'azione, e si occupa di compiere gli step successivi.

Esempio: nella view viene lanciato un evento: il giocatore ha scelto di pescare una carta. A questo punto il controller, in ascolto di possibili eventi, grazie all'evento lanciato, riconosce che è avvenuta questa azione, e si occupa di controllare (appunto *controller*) che l'azione sia valida. Interroga quindi il model, verificando che il giocatore abbia un numero di carte minore di 30 (limitazione spiegata nella 1), e nel caso il model restituisca un qualche valore che consente l'azione, allora l'azione viene correttamente effettuata, altrimenti si avvisa il giocatore che l'azione è *illegale*.

2.2 Model

Il model contiene tutti i metodi contenenti la logica effettiva del gioco. La classe principale che si occupa di contenere il model è

```
public class GameModel
```

A livello "logico", è naturale che nel model del gioco uno siano presenti le seguenti cose:

- Il mazzo di carte, implementabile in Java da una lista. Abbiamo usato un ArrayList.
-

```
    // Mazzi di carte usate
        private CardDeck cardsDeck;
        private UsedPile usedCards;
```

```
    // ArrayList contenente i giocatori
        private ArrayList<Player> players;
        private int numberOfPlayers;
        private int maxNumberOfPlayers;

    // Gestore dei turni
        private PlayerRoundIterator turnIterator;
        private CardColor currentCardColor = null;
```

2.3 View

2.4 Controller