JUNO

D’annibale Luigi

N.Matricola : 1991254

Corso: canale M-Z presenza

Gruppo: Venturini Daniele (1991255)

Indice

- Compito assegnato

Considerazioni

- Scelte iniziali sull'utilizzo dei pattern
- Il modello
- L'interfaccia grafica

Implementazione del modello

- Il package "Cards"
- Il package "Rules"
- Il package "Players"

Indice

Implementazione dell'interfaccia grafica

- Il package "Animation"
- Il package "Elements"
- Il package "Pages"
- Il package "Pages.ProfilePanels"

Implementazione del controllo

- Il package "Utilities"

Il compito assegnato:

Progettare e sviluppare la versione giocabile in Java del gioco UNO.
- in gruppo di 2 (con lo sviluppo obbligatorio di 3 modalità diverse di gioco, e di animazioni ed effetti speciali, effetti audio) (specifiche da 1 a 8)

1) Gestione del profilo utente, nickname, avatar, partite giocate, vinte e perse, livello ...
2) Gestione di una partita completa in modalità classica con un giocatore umano contro 3 giocatori artificiali
3) Uso appropriato di MVC [1,2], Observer Observable e di altri Design Pattern
4) Adozione di Java Swing [2] o JavaFX [3] per la GUI
5) Utilizzo appropriato di stream
6) Riproduzione di audio sample (si veda appendice AudioManager.Java)
7) Animazioni ed effetti speciali
8) Almeno altre 2 modalità oltre a quella classica

Considerazioni iniziali:

Scelta di utilizzo del MVC pattern.

È stato scelto di utilizzare il pattern Model-View-Controller per strutturare il codice in modo da separare interamente la gestione dell'interfaccia utente, e la gestione del codice esclusivamente legato al modello del gioco.

L'interfaccia utente infatti oltre a mostrare le viste di gioco deve occuparsi anche di gestire l'esperienza di gioco dell'utente e deve essere tarata in base all'utente finale,

invece il modello ha un compito molto più astratto e generico, gestire partite del gioco di carte Uno.

Astrarre il modello, ha come obiettivi principali il rendere il codice riutilizzabile, e di renderlo indipendente dall'interfaccia grafica associata.

Considerazioni iniziali:

Scelta di utilizzo del Observer-Observable

È stato scelto di associare al pattern MVC, il pattern Observer-Observable, quindi modellare il tavolo di gioco come osservabile e poi andare a registrare la vista come osservatrice.

Questo consente la comunicazione diretta dal tavolo di gioco alla vista mentre la comunicazione tra vista e tavolo di gioco è più articolata, cioè gli eventi in ingresso della vista vengono registrati dal controllo che ne delega la gestione al tavolo di gioco.

Considerazioni sul modello:

Astrazione.

Se l'obiettivo principale dell'astrazione del modello è il principio di riuso, individuare i compiti principali e separarli era la chiave di volta per rendere il codice mantenibile e aggiornabile.

Il primo compito fondamentale è quello del regolamento della partita, che definisce chiaramente le tre fasi di una partita : la preparazione del gioco, il gioco stesso, e la fine del gioco. La delegazione di questo compito consentirà di gestire non una, bensì diverse modalità di gioco benché queste abbiano un regolamento.

La fase iniziale, e quella finale sono più facili da definire rispetto a quella del gioco che potrebbe richiedere maggiore attenzione.

Considerazioni sul modello:

Astrazione e caso reale.

Analizzando i comportamenti che avvengono durante una partita, due punti importanti sono: la gestione del singolo turno, e l'interazione che avviene tra giocatori, mazzo di gioco e pila degli scarti.

Durante lo svolgimento di un singolo turno, potrebbero verificarsi diversi eventi:

è stato deciso di delegare la gestione degli eventi riguardanti il flusso del turno ad un gestore di turno, che tenga traccia dell'ultima carta giocata, dei giocatori, del loro turno e del verso in cui questo si muova,

e di delegare la gestione del transito delle carte ad un gestore di mazzo, incaricato anche della pila degli scarti.

Considerazioni sul modello:

Astrazione e caso reale.

La partita si svolge quindi, su un tavolo di gioco, a cui sono assegnati

- dei giocatori,
- un gestore di regole,
- un gestore di turno,
- un gestore di mazzo.

Sarà quindi il tavolo di gioco il nostro modello, colui che andrà a gestire una o più partite di Uno, che saranno personalizzate in base al regolamento che affideremo al gestore di regole;

il modello infatti dovrà essere dinamico e reattivo alle scelte dell'utente, indipendemente da quale interfaccia grafica queste vengano reperite. Dovrà quindi aggiornarsi in base alle scelte, gestirle e continuare il gioco aspettandone di nuove.

Considerazioni sul modello:

Astrazione e dati.

La profilazione che il modello deve prevedere è semplice, e riguarda il salvataggio e l'aggiornamento delle statistiche di gioco, quale il numero di partite vinte e perse e l'esperienza(livello) che i vari giocatori accumulano giocando, questo tipo di dati deve essere gestito dal modello, e non è delegabile all'interfaccia grafica.

Quindi per separare il livello di accesso ai dati è stato introdotto un gestore di giocatori che si occupi della loro gestione.

Considerazioni sull'interfaccia grafica:

Profilazione utente.

Innanzitutto la profilazione utente è un compito che dipende molto dal contesto, perciò si presta più ad essere gestita dall'interfaccia grafica, o almeno in gran parte, infatti un giocatore non sempre corrisponde ad un utente dal profilo complesso, e le informazioni che compongono il profilo di un utente sono fortemente collegate al contesto dell'applicativo.

Gli aspetti di maggiore rilevanza per quanto riguarda i dati utente sono :

- le impostazioni dell'utente
- le statistiche delle partite dei giocatori
- la loro autenticazione

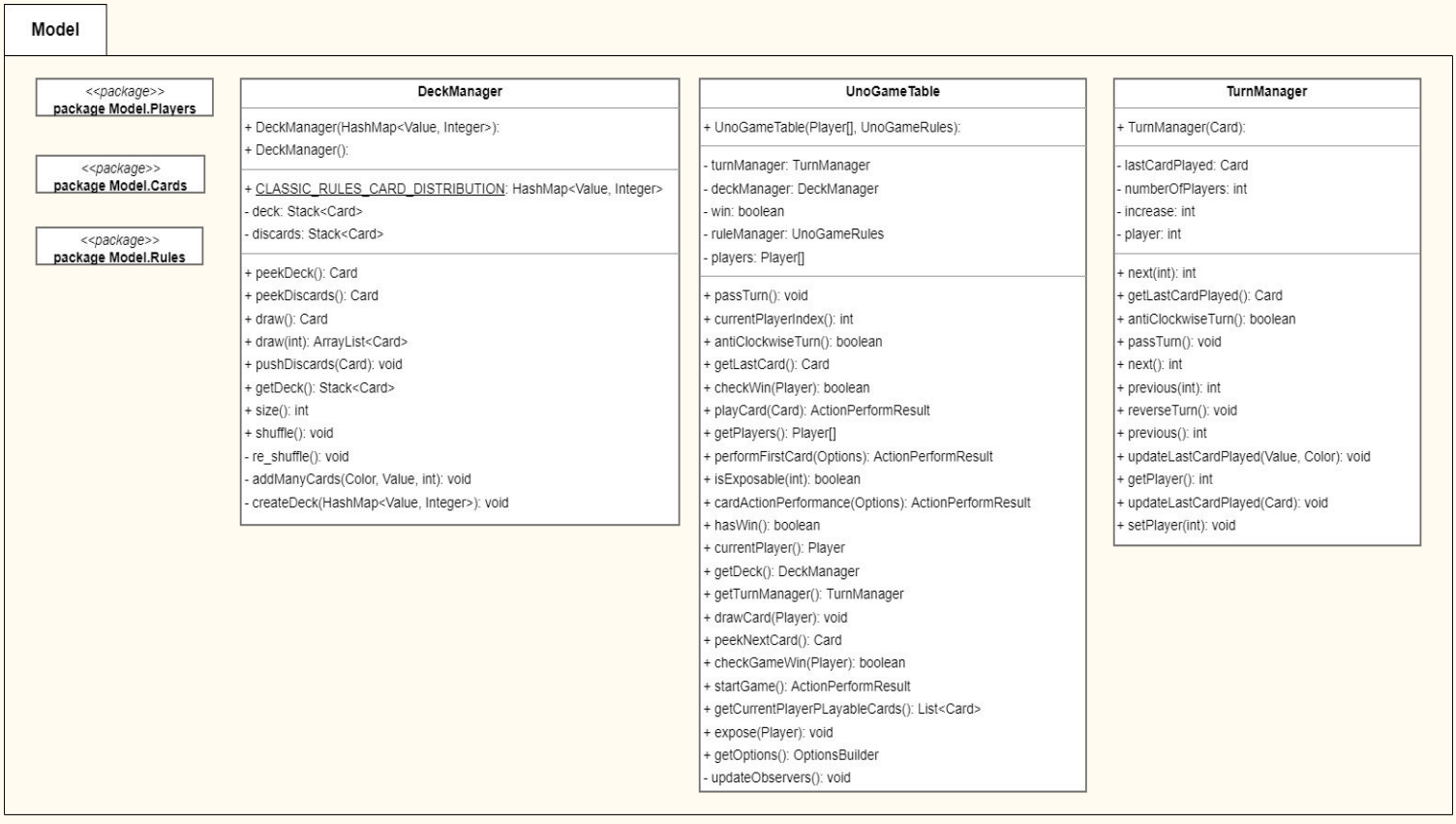
Considerazioni sull'interfaccia grafica:

Personalizzazione dell'ambiente di gioco.

L'interfaccia utente deve essere facile da usare e prevedere effetti grafici ed audio, che ovviamente l'utente deve poter personalizzare creando così le impostazioni più adatte alle sue esigenze.

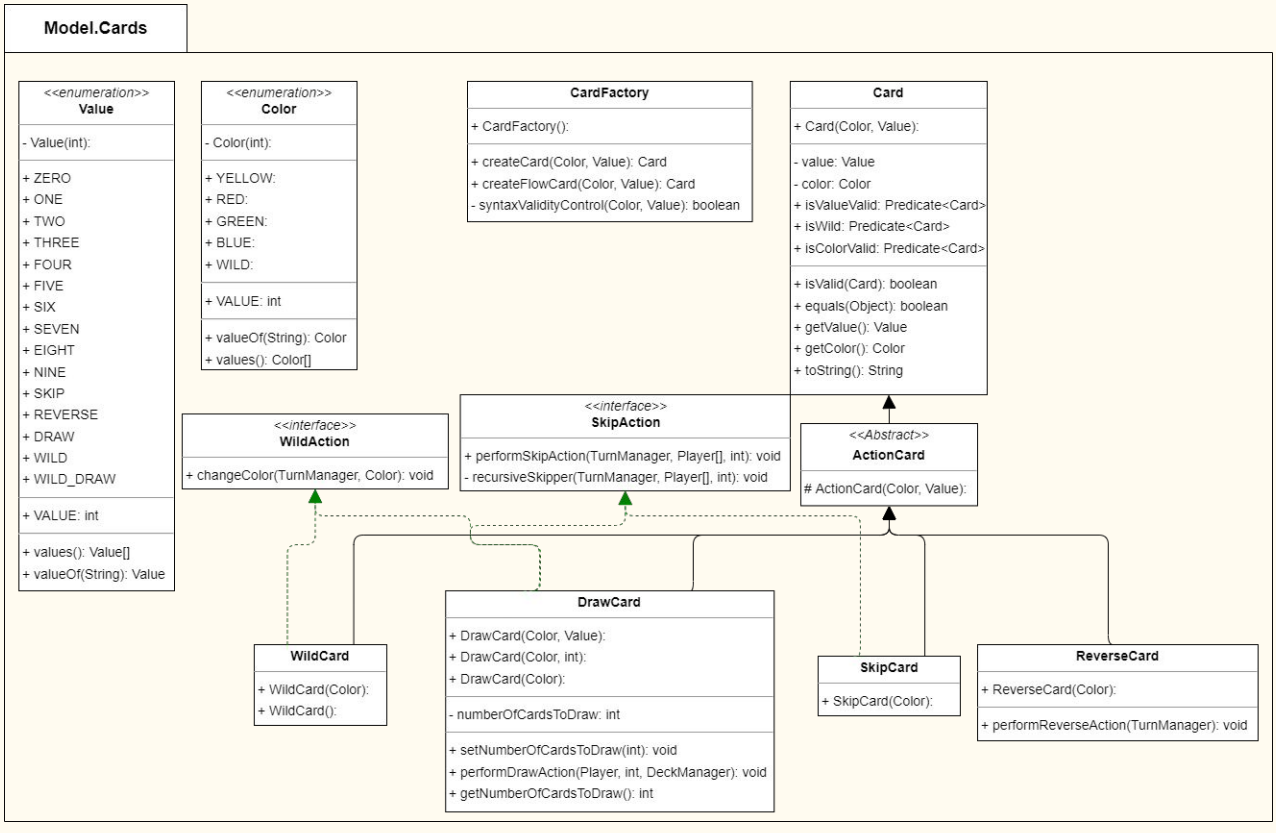
Le impostazioni servono a migliorare l'esperienza di gioco, ed una volta gestite queste ed i dati utente, la cosa fondamentale è creare una vista adatta al modello, facile da utilizzare per l'utente, che deve riuscire ad entrare nell'ambiente di gioco, e interagire con gli altri giocatori per concorrere alla vittoria del gioco.

Implementazione del modello: *il "model"*



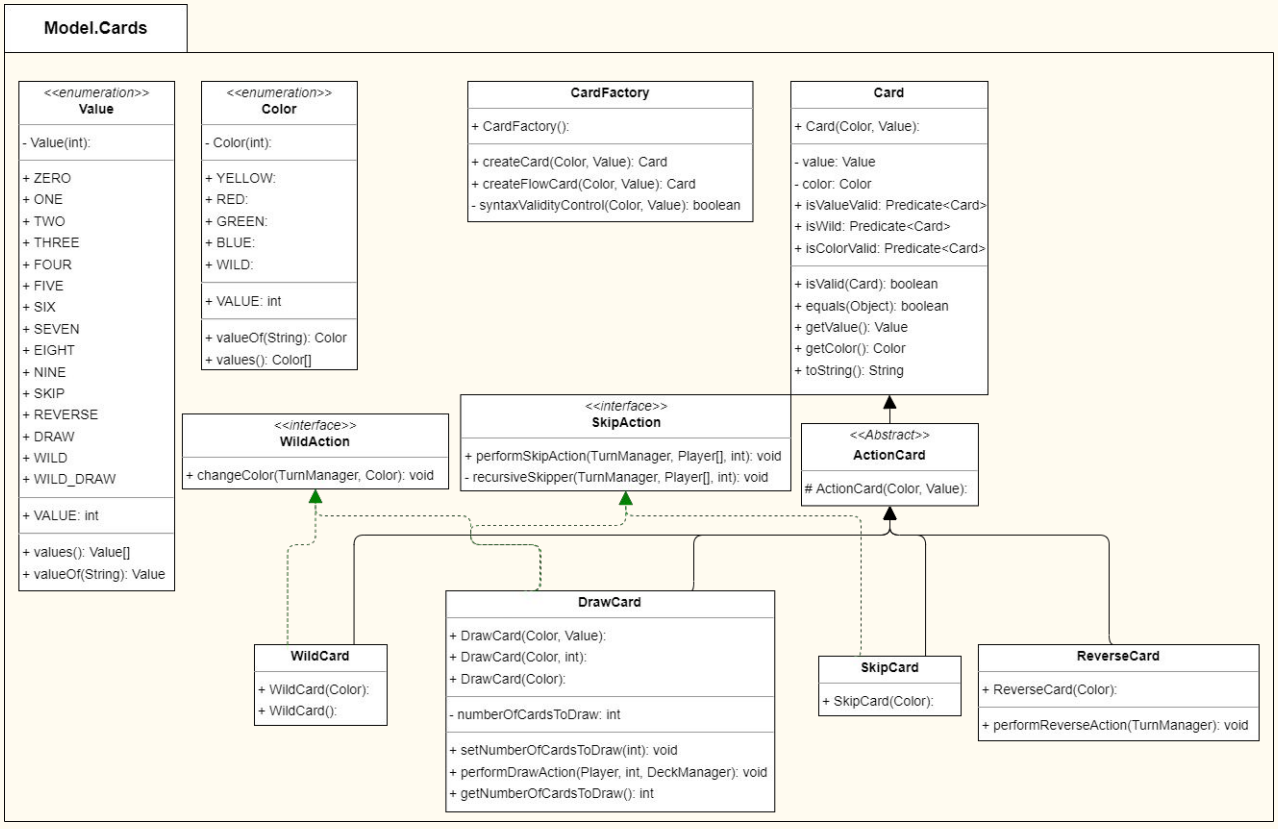
È stato implementato un tavolo di gioco "UnoGameTable", che utilizza un gestore di turno "TurnManager" e un gestore di mezzo "DeckManager".

Implementazione del modello:
il package delle carte



Sono state definite le enumerazioni “Value” e “Color” per identificare i valori ed i colori delle carte.
È stata utilizzata l’ereditarietà per definire le varie carte, “Card” rappresenta la carta più semplice possibile con un valore, ed un colore.
La “ActionCard” che estende “Card” rappresenta una carta azione, ovvero una carta a cui è associato un effetto.

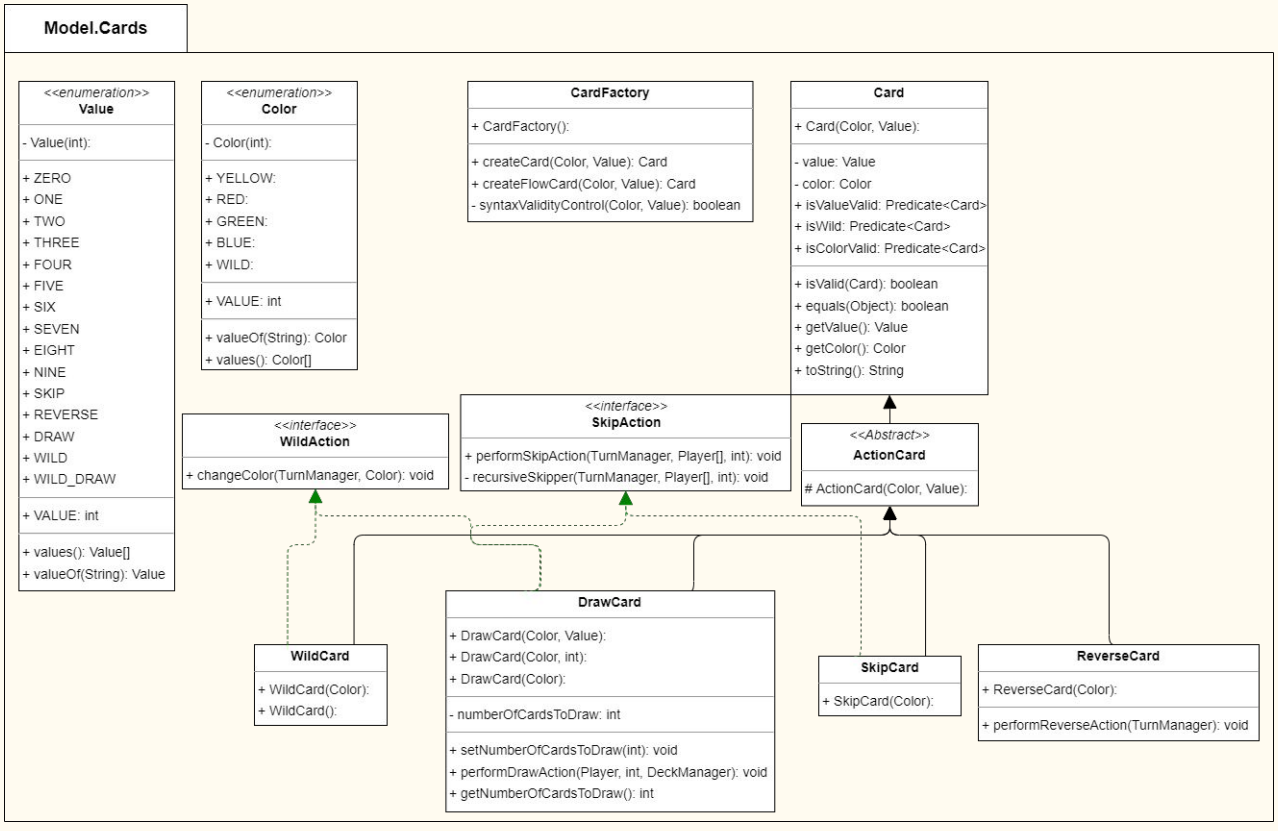
Implementazione del modello:
il package delle carte



Gli effetti previsti da “ActionCard” vengono definiti dalle classi che la estendono, ovvero :

- “SkipCard”, “DrawCard” che implementano l’interfaccia “SkipAction” la prima rappresenta la carta blocco, che fa saltare il turno; la seconda imlpementa anche “WildAction” e rappresenta la carta pesca che fa pescare un numero di carte, 2 oppure 4 e si sceglie un colore, e fa saltare il turno.
- “ReverseCard” rappresenta la carta cambio giro.
- “WildCard” che implementa “WildAction” rappresenta la carta cambio colore.

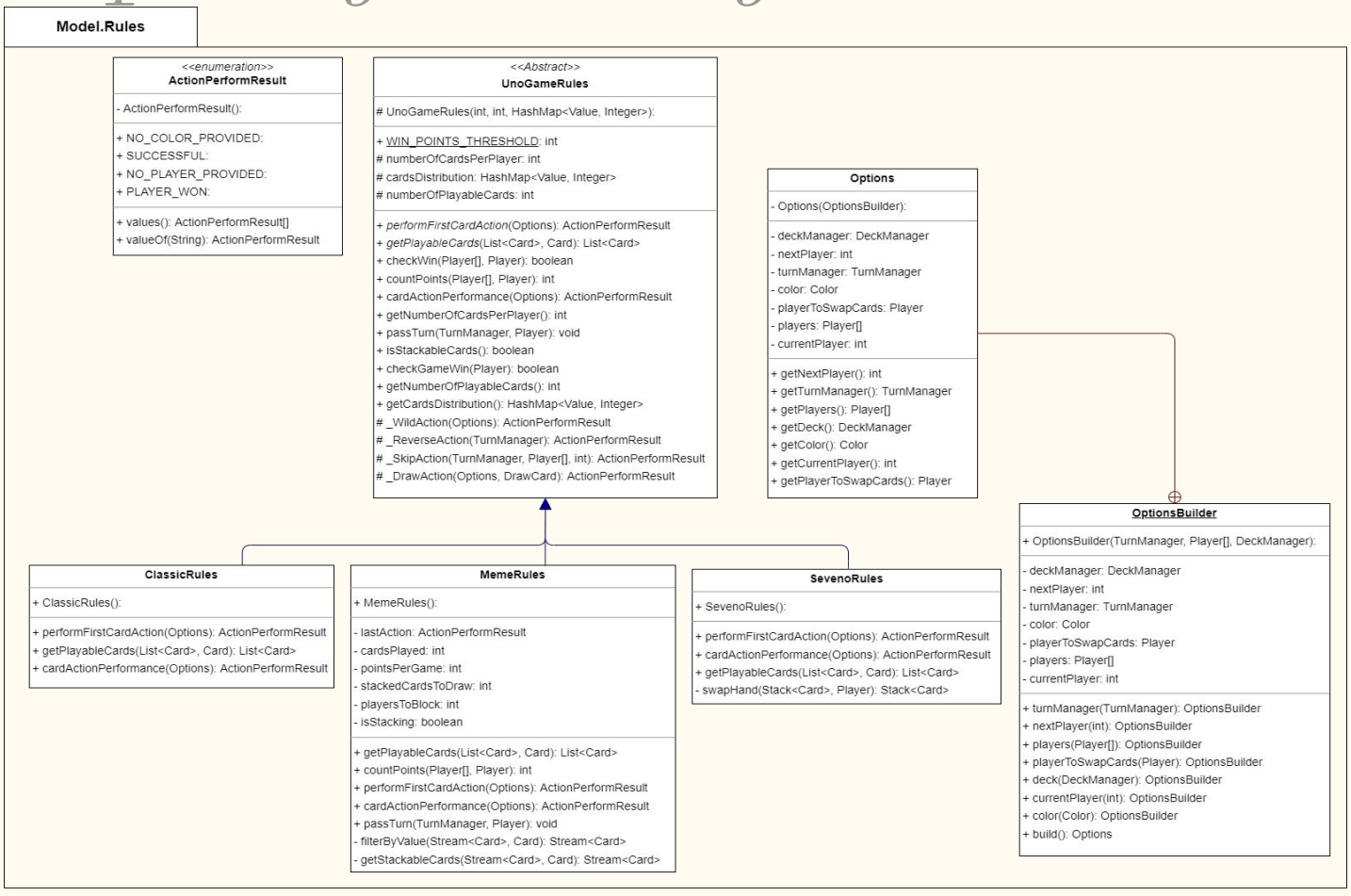
Implementazione del modello:
il package delle carte



Le carte utilizzano il pattern di costruzione SimpleFactory, sono previste 2 modi per la creazione di carte:

- creazione di carte normali: le quali sono soggette alle regole di validità del gioco Uno, che non prevede alcune combinazioni tra colore e valore (come ad esempio un cambio colore rosso),
- creazione di carte di flusso: che non sono soggette a regole.

Implementazione del modello:
il package delle regole

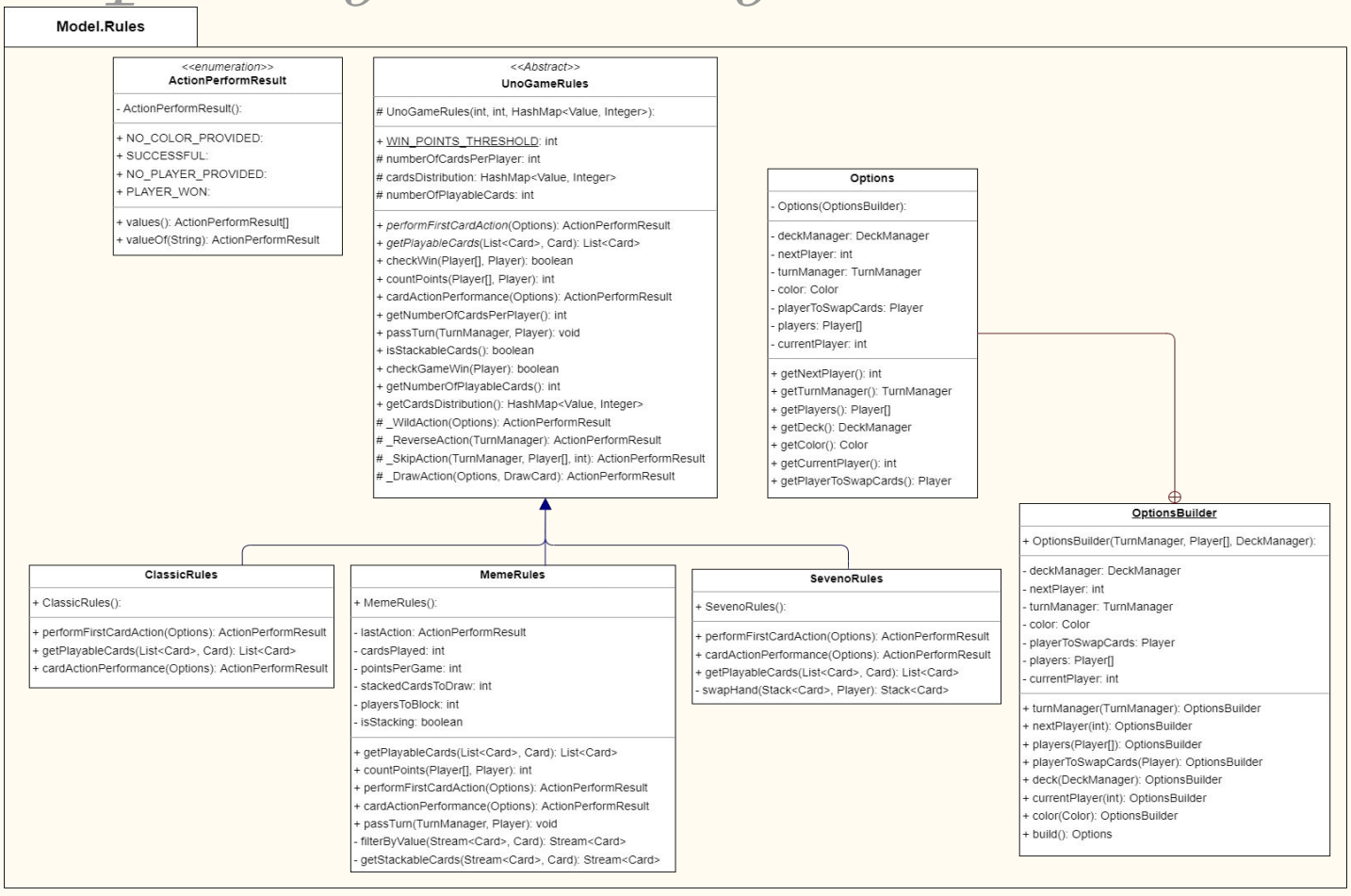


"UnoGameRules" è il gestore di classi astratto che viene esteso da tutto i gestori di regole concreti.

I gestori di regole quindi devono tutti definire:

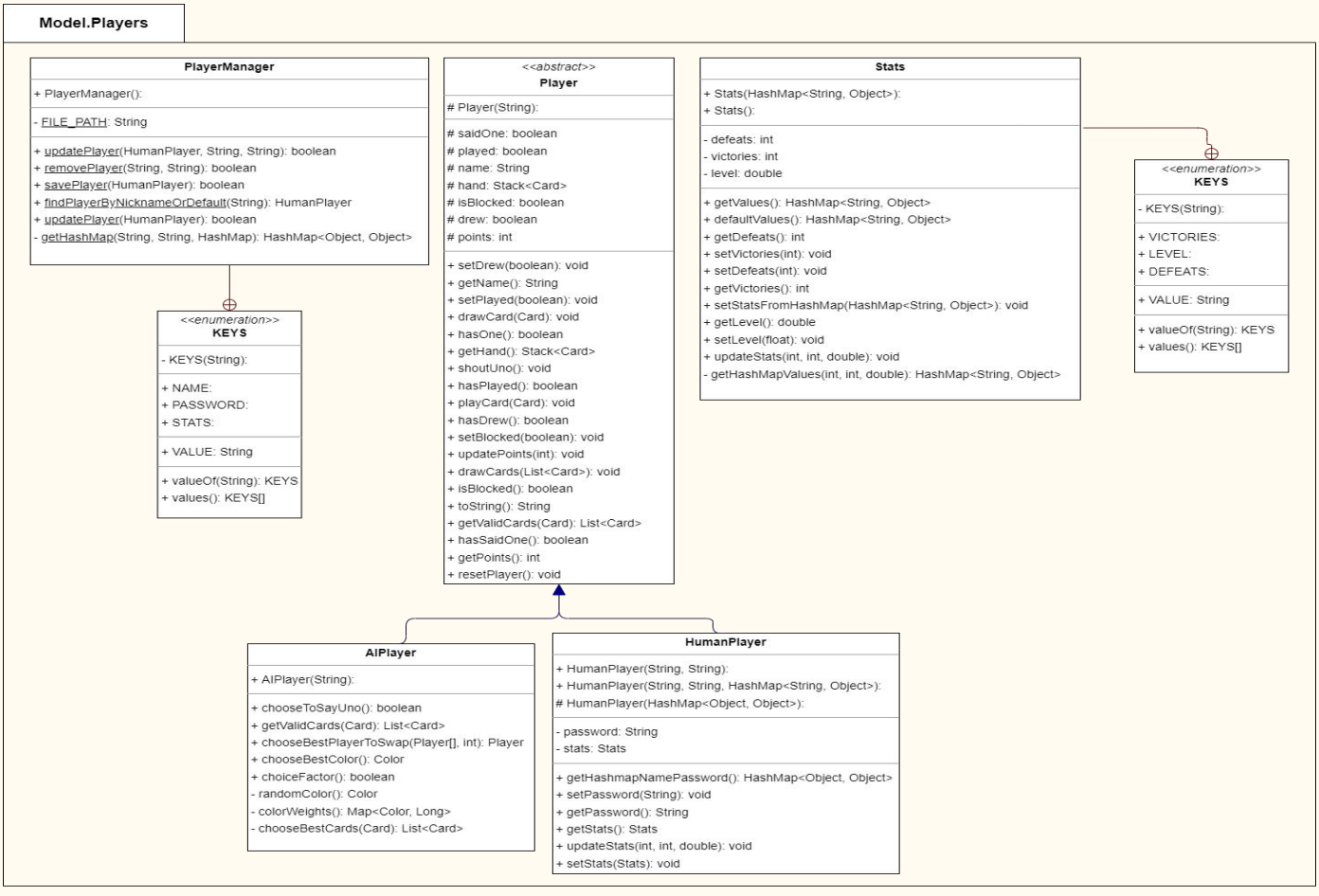
- il numero di carte distribuite all'inizio del gioco
- il numero di carte giocabili per turno da un giocatore
- come è costituito il mazzo di gioco
- il concetto di giocabilità di una carta
- gli effetti delle carte azione
- la condizione di vittoria
- come viene effettuato il conteggio dei punti

Implementazione del modello:
il package delle regole



È stata dichiarata la enumerazione “ActionPerformResult”, che fornisce l’elenco degli eventi che possono avvenire successivamente all’effetto di una carta azione.
Viene utilizzato il pattern di costruzione Builder per costruire degli oggetti “Options”, il quale compito è racchiudere i parametri dei metodi del gestore di regole che sono astratti, infatti alcune modalità hanno bisogno di parametri personalizzati, e in futuro nuove modalità potrebbero necessitare di nuovi, gli oggetti “Options” incapsulando i parametri al loro interno, risolvono la problematica.

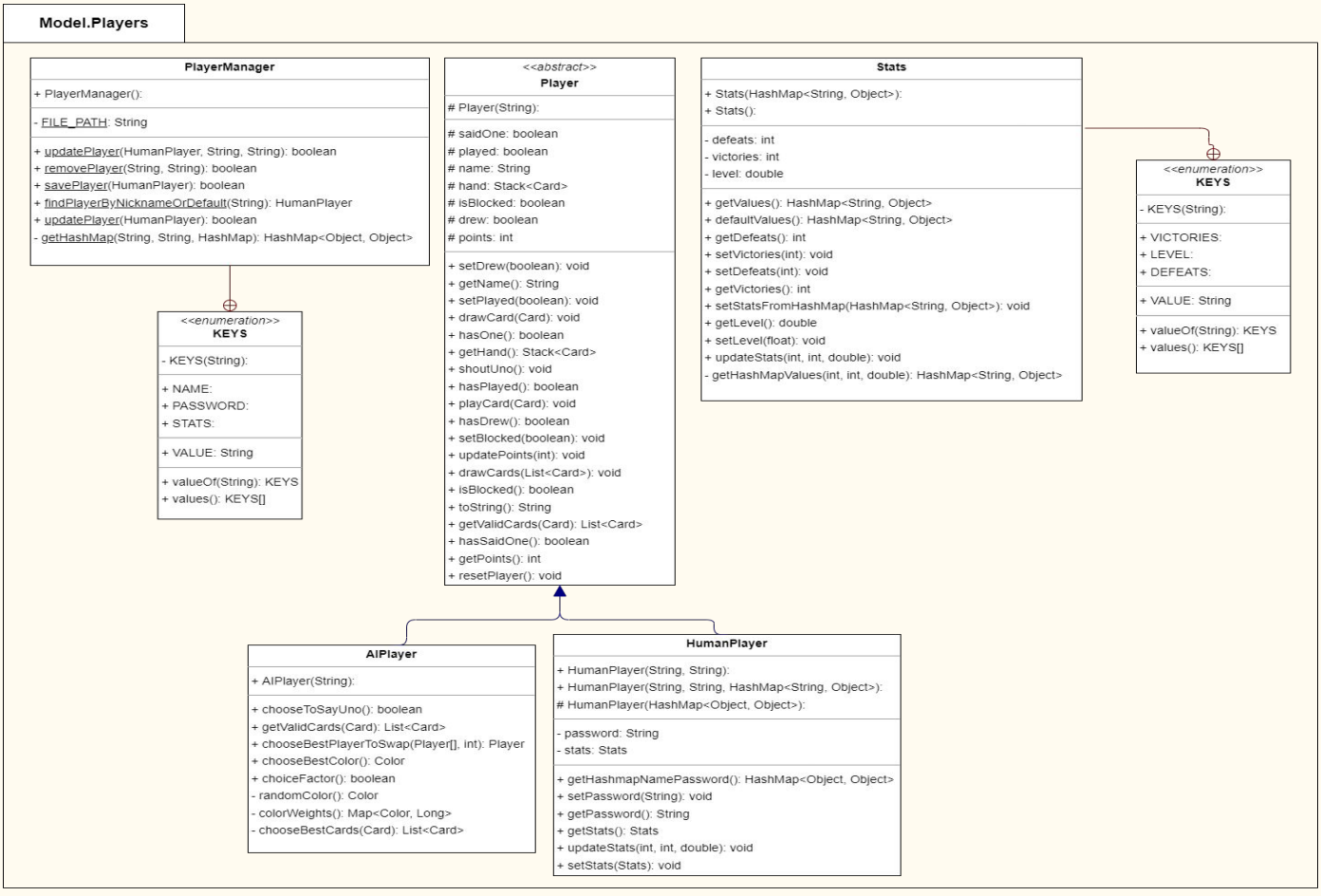
Implementazione del modello:
il package dei giocatori



È stata dichiarato un gestore di giocatori “PlayerManager” che utilizza una libreria (org.json - javadoc) per scrivere su file, in notazione json, che si occupa di verificare, salvare, aggiornare e rimuovere i dati del giocatore dai file, utilizzati come base di dati.

È stata dichiarata la classe che modella i dati di un giocatore “Stats” che lavora con il gestore di giocatori e salva partite vinte, perse ed il livello del giocatore.

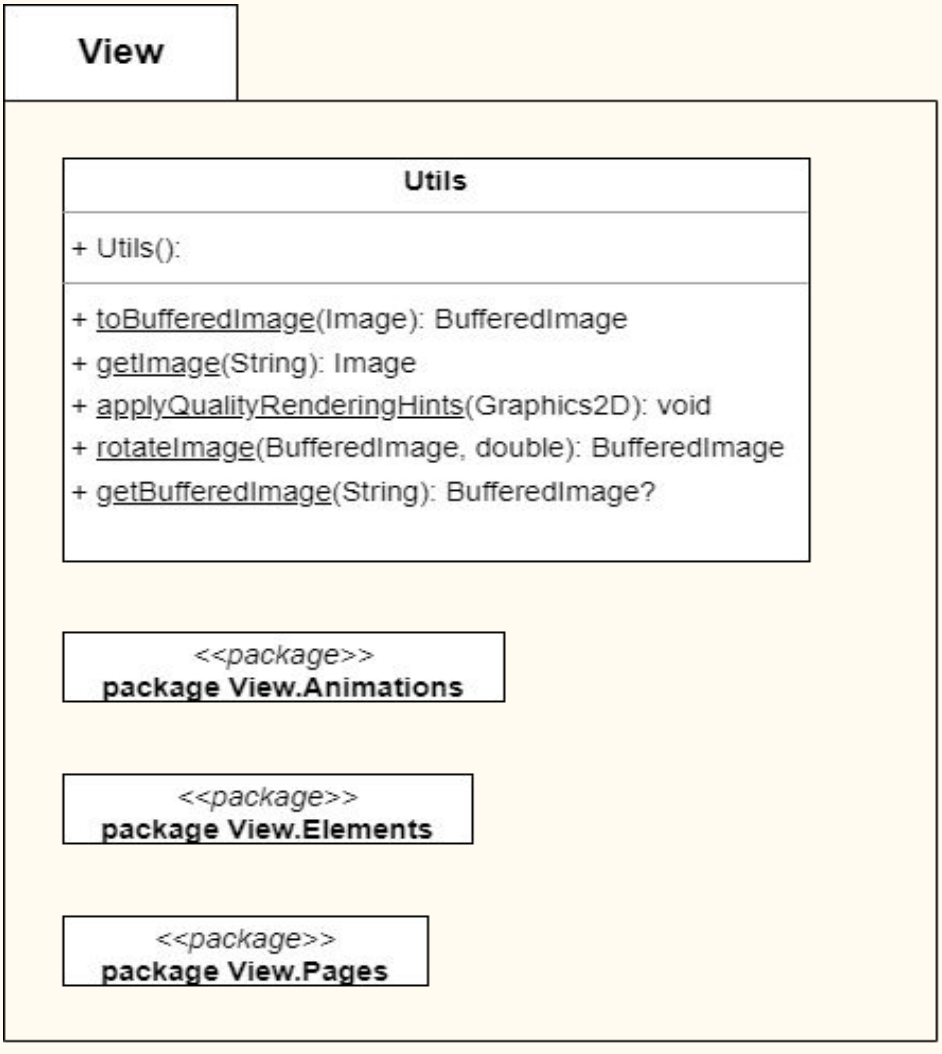
Implementazione del modello:
il package dei giocatori



È stato modellato il concetto di giocatore attraverso la classe astratta “Player” che fornisce tutti gli attributi e le funzionalità di un giocatore di base, questa viene estesa da:

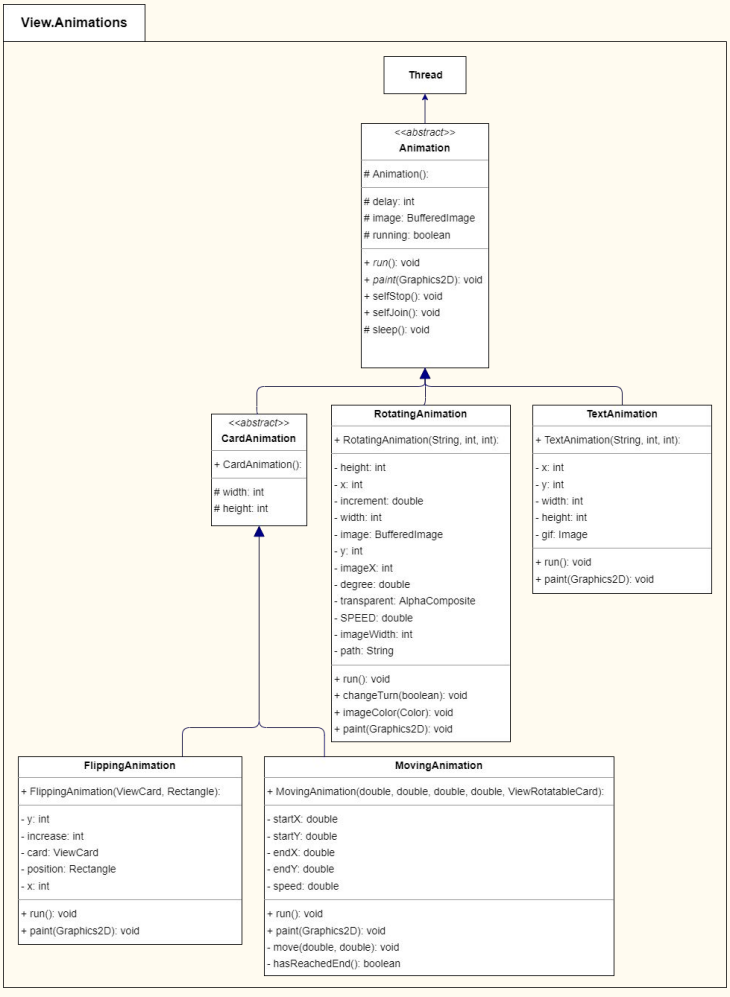
- “HumanPlayer” che rappresenta un giocatore umano,
- “AIPlayer” che rappresenta un giocatore controllato da intelligenza artificiale, il “fattore umano” che può essere lo scegliere quale carta giocare, oppure quale colore scegliere, oppure dimenticare di dire UNO, oppure non accorgersi che qualcun altro non l’ha detto, viene tutto simulato attraverso metodi specifici, per rendere questi giocatori “intelligenti” ma allo stesso tempo capaci di sbagliare.

Implementazione dell'interfaccia grafica : *la "view"*



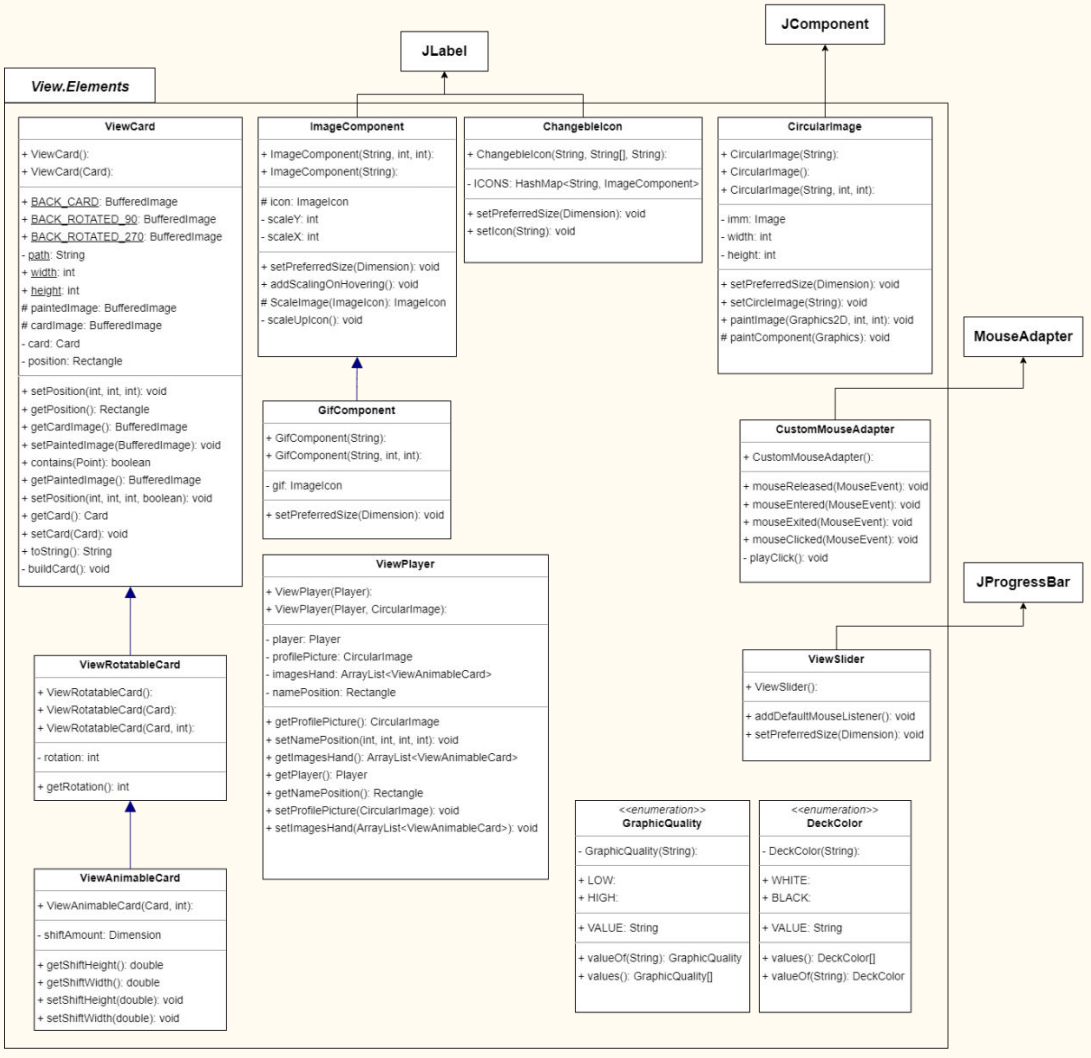
"Utils" è una classe con metodi statici, che vengono utilizzati in tutto il package dai vari pannelli per applicare migliorie grafiche oppure richiamare metodi molto frequenti.

Implementazione dell'interfaccia grafica : *il package delle animazioni*



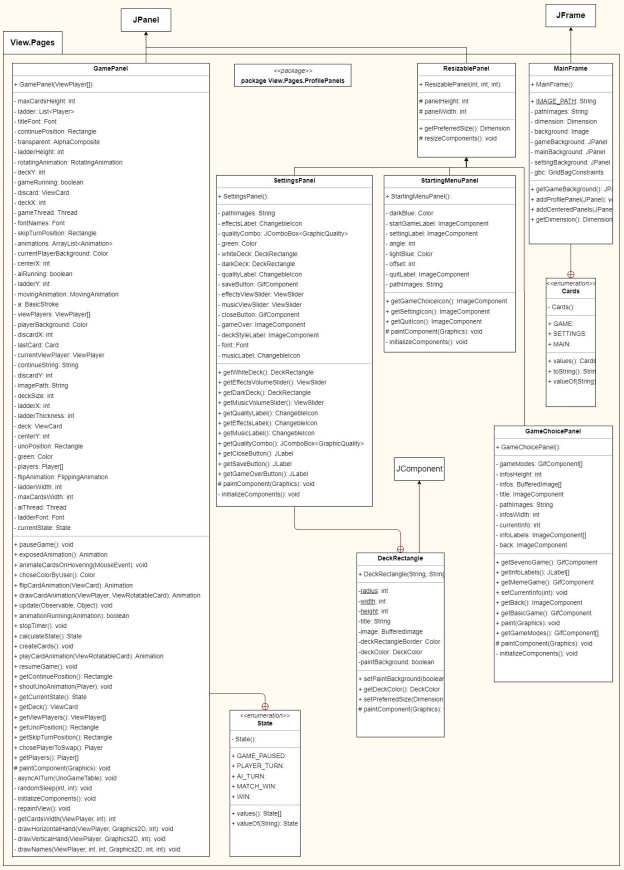
Le animazioni servono a rendere migliore l'esperienza di gioco, sono effetti visivi e vengono applicati a delle immagini.
"Animation" è la classe astratta che ne ingloba i comportamenti generali, poi queste si suddividono in vari tipi : "TextAnimation" è l'animazione testuale si occupa di animare dei testi, "RotationAnimation" è l'animazione di rotazione e si occupa di occupa di disegnare gli indicatori del turno, grandi frecce che ruotano nel verso del turno, ed hanno il colore dell'ultima carta giocata.
"CardAnimation" è astratta ed aggiunge alla superclasse proprietà per animare immagini di carte, che vengono ereditate da:
"FlippingAnimation" l'animazione di girata, serve a scoprire la faccia di una carta coperta,
"MovingAnimation" l'animazione di movimento, che muove carte da una parte all'altra nelle viste.

Implementazione dell'interfaccia grafica : *il package degli elementi*



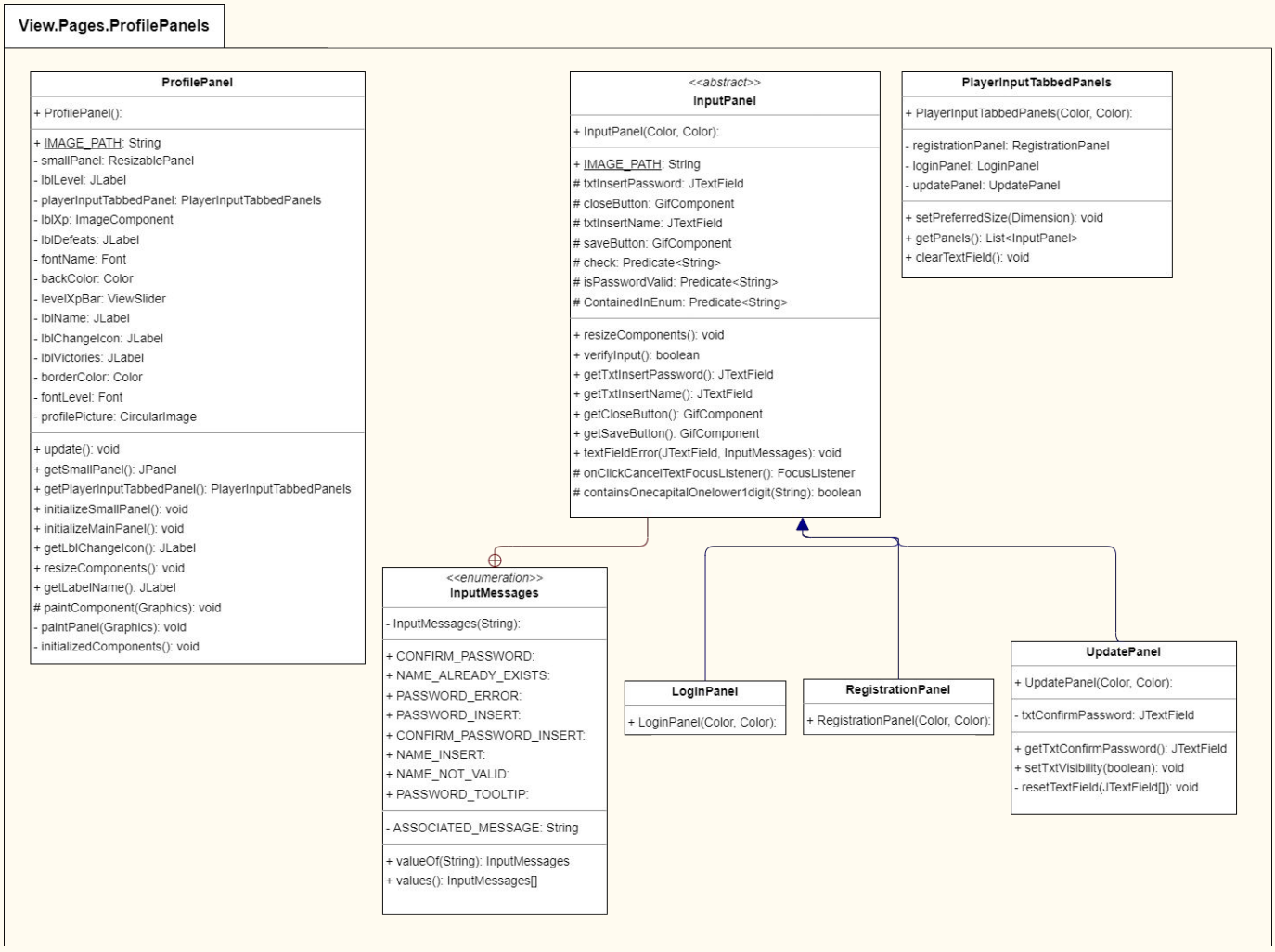
In questo package sono contenuti gli elementi grafici che vengono aggiunti ai panelli.

Implementazione dell'interfaccia grafica : *il package delle pagine*



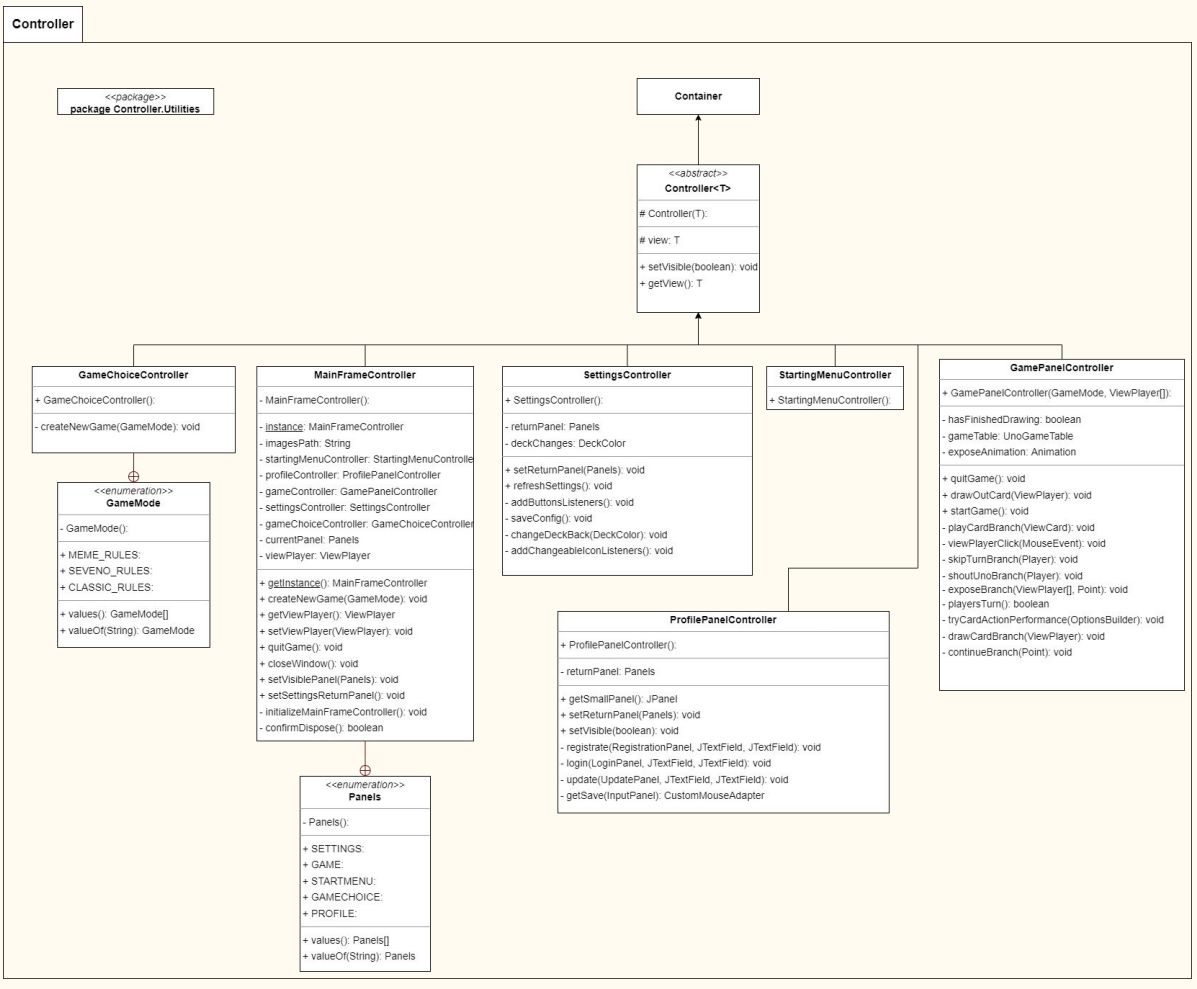
Quivi sono contenute tutte le pagine dell'interfaccia grafica ovvero:
"MainFrame" che estende "JFrame" che funge da contenitore per tutti gli altri pannelli,
"StartingMenuPanel" (ridimensionabile) è la schermata iniziale di scelta,
"SettingsPanel" (ridimensionabile) è la schermata delle impostazioni da cui è possibile cambiare il colore delle carte nel gioco i volumi di musica ed effetti e la qualità grafica,
"GameChoicePanel" (ridimensionabile) è la schermata di scelta della modalità di gioco,
"GamePanel" che estende "JPanel" ed è la schermata di gioco, la sua grandezza è fissa ed è a schermo intero.
Inoltre "ResizablePanel" rende ridimensionabili i pannelli che lo ereditano, dove per ridimensionabili si intende che i pannelli che lo ereditano ridimensionano i lori componenti interni in base alla risoluzione corrente dello schermo.

Implementazione dell'interfaccia grafica : *il package dei pannelli del profilo*



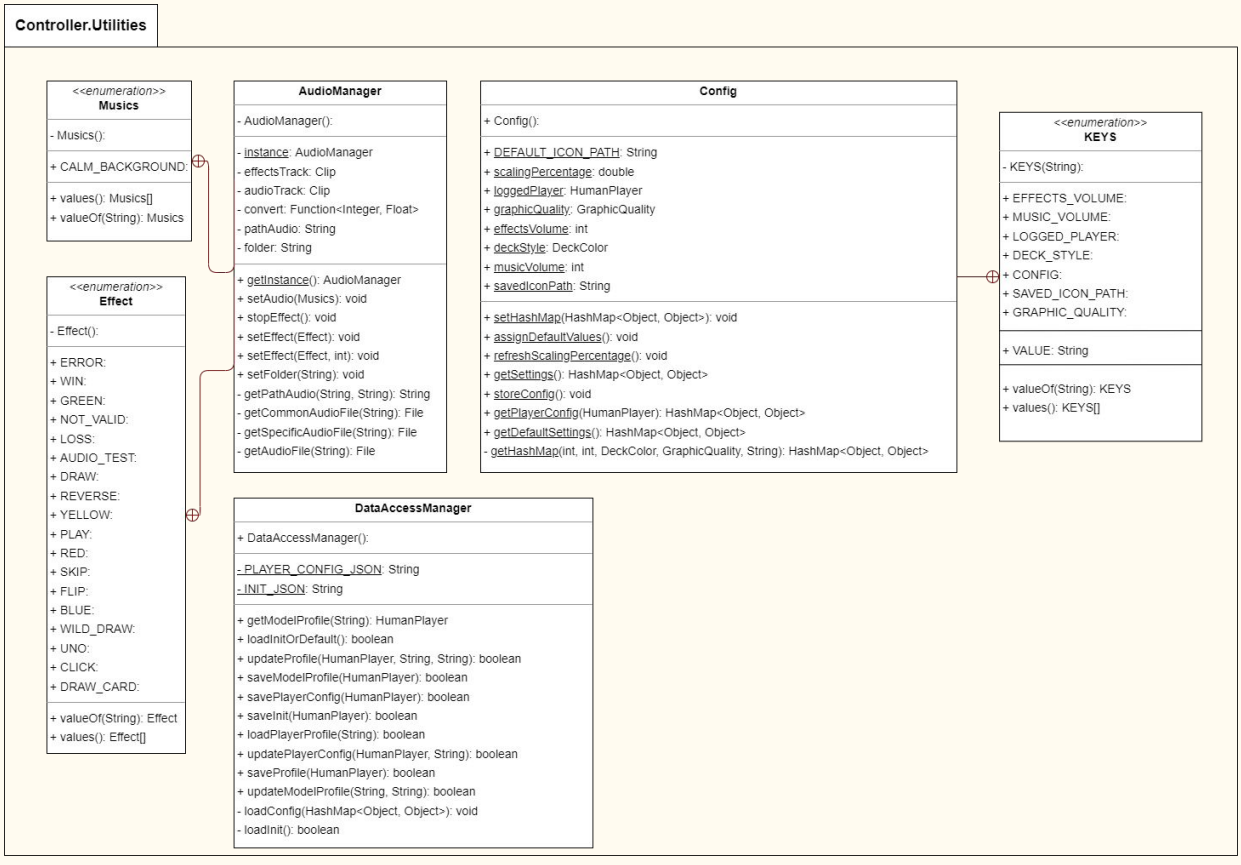
Il pannello del profilo "ProfilePanel" ha la versione piccola in alto sovrapposta ad altri pannelli oppure la versione grande al centro del frame per l'accesso ai dati:
"InputPanel" classe astratta che rappresenta un pannello di input provvede a fornire elementi di inserimento con i quali l'utente puo inserire le credenziali, le specializzazioni vengono fatte per fornire ad ogni pannello le giuste proprietà:
"UpdatePanel" permette di aggiornare i propri dati,
"LoginPanel" permette di effettuare l'accesso,
"RegistrationPanel" permette di creare un nuovo utente.

Implementazione del controllo:
il “controller”



Il controllo è ripartito in vari controlli, uno per vista, questi aggiungono “ascoltatori di eventi” e rilevano gli eventi che l’utente scatena come il click del mouse in un punto dello schermo. Collegano ai rispettivi eventi i metodi delle viste oppure del tavolo di gioco che devono gestirli.
È stato implementato il pattern di progettazione Singleton sul “MainFrameController” ovvero il controllo della finestra principale, ciò poiché deve essere istanziato una sola volta e tutti i controlli devono relazionarsi con lui.

Implementazione del controllo:
il package delle utilità



Contiene tutte le utilità di cui il controllo ha bisogno, inoltre contiene il gestore di audio “AudioManager” che si occupa di riprodurre musica ed effetti su richiamo del controllo, “Config” la classe delle impostazioni ha attributi e metodi statici, infatti questa viene utilizzata come contenitore e non è utilizzata solo dai controlli ma anche dalle viste.
Inoltre è stato dichiarato il gestore per l’accesso ai dati “DataAccessManager”, che va a lavorare con il livello di accesso ai dati del gestore di giocatori ma non solo, prevede salvataggio ed aggiornamento delle impostazioni personali dell’utente.