# JTRASH

- Numero Matricola :1972910
- Corso : Presenza M/Z
- Nome: Gianmarco Mottola
- Decisioni di progettazione:
    1. Per la gestione del profilo utente ho pensato di usare un file txt come "database" dove scrivo ogni volta che l'observer della view del profilo viene notificato e leggo ogni volta che carico la scena (frontend). La logica di modifica/conferma l'ho pensata in termini di visibile/non visibile. Per gli avatar ho utilizzato un oggetto con immagini e che costruisco tramite path.
    2. Per la gestione della partita contro giocatori artificiali ho pensato di automatizzare il bot controllando (per ogni carta) che la carta da giocare (cioe quella pescata per il primo turno e quella scambiata dalla mano dal secondo in poi) abbia una posizione valida nella mano del bot e che il jolly (King) si inserisca nella prima posizione valida trovata e che a fine turno di ogni bot controllo se ha vinto. Mentre il giocatore reale viene condizionato dagli eventi, decide lui che carta scambiare, quando finire il turno o se controllare la vittoria del round/partita. Il FrontEnd viene costruito nella View controllando quanti giocatori giocano (variabile passata dal profilo) e costruendo le mani e i componenti (GridPane) con un unico ciclo. Come nel profilo, nella View viene aggiornato il backend ogni volta che il giocatore reale scambia una carta.
    3. Model: per gli oggetti utilizzati per accedere ai "dati" del gioco -> Observable. View: per visualizzare e interagire con l'utente -> Observer. Controller: per manipolare gli eventi del frontend(View). Ho utilizzato il singleton per i controller.
    4. Ho utilizzato JavaFX con SceneBuilder per la GUI perche mi sembrava piu completo e personalizzabile di Java Swing
    5. Ho utilizzato gli stream per scrivere codice piu compatto, principalmente come operazione terminale (forEach)
    6. Ho utilizzato l'audio quando inizia la partita e per ogni scambio carta del giocatore reale
    7. Ho utilizzato le animazioni per ogni scambio di carta dei bot

- Ho usato MVC per distinguere tra back (Model) e front end (View) e ho utilizzato il Controller per manipolarli. Il model è un Observable mentre la view è un Observer. Ho utilizzato il Singleton per garantire che ci fosse una sola instanza dei controller.
- Non ho trovato un utilizzo specifico per loro, ma li ho aggiunti una volta terminato il progetto per rendere il codice piu compatto.
-