Relazione di Progetto BlackJack

Enrico Baroni, Alberto Rossi

Ottobre 2021

Indice

Capitolo 1
Analisi

Il BlackJack è un gioco di casinò più famoso al mondo, si svolge tra il giocatore e il banco che è rappresentato dal dealer.
Una volta che il giocatore ha fatto la sua puntata, il dealer procede nel distribuire le carte procedendo da sinistra a destra e assegnando l'ultima a se stesso. Effettuato il primo giro procede così con il secondo, ma la carta che si assegna è coperta.
Una volta assegnate le carte il dealer procede nel leggere il punteggio e aspetta le giocate del player che può chiedere carta o stare, a sua discrezione.
Una volta finito il turno del giocatore il dealer si appresta a sviluppare il proprio gioco seguendo le regole del banco: egli deve chiedere carta se ha un punteggio inferiore a 17, stare se ha un punteggio pari o superiore a 17 e se supera il punteggio di 21 il dealer "sballa" e deve pagare il giocatore rimasto.
Una volta finito il turno del dealer si procede nel confrontare i punteggi pagando le combinazioni superiori alla sua, ritirando le combinazioni inferiori alla sua e lasciando sul banco le combinazioni uguali alla sua.
Infine se il giocatore effettua un 21 con le prime due carte assegnate dal dealer, ricevendo quindi un asso e una figura si forma così il "blackjack" e ha diritto al pagamento di 3 a 2.
Il gioco può andare avanti all'infinito o finché il giocatore ha finito il budget a disposizione.

1.1 Requisiti

L'applicazione è realizzata come elaborato di "Programmazione ad Oggetti " del corso di Ingegneria Scienze Informatiche dell'Università di Bologna.

Requisiti funzionali

- interfaccia di gioco
- gestione chips di gioco
- implementazione Ai dealer
- implementazione single Player vs Ai(dealer)

Requisiti non funzionali

- multiplayer
- animazione movimento carte

1.2 Analisi e modello del dominio

L’applicazione all’inizio fornisce la possibilità di fare la propria “puntata” all’utente e una volta eseguita inizia il gioco.

L’applicazione pesca automaticamente le prime due carte del giocatore e del dealer (nascondendo però la seconda carta del dealer all’utente), dopo di che si sbloccheranno i tasti all’utente che gli permettono di eseguire la propria giocata (chiedere carta o stare).

Una volta che l’utente ha finito il proprio turno il banco “apre” cioè esegue il proprio turno scoprendo la carta tenuta coperta ed eventualmente pescarne ancora se il suo punteggio è inferiore a 17.

Dopodichè finito sia il turno del player sia il turno del dealer si confrontano i due punteggi pagando l'eventuale vittoria del player.

Finito il budget viene data la possibilità al player di ricominciare una nuova partita con il budget iniziale oppure di uscire dal gioco.

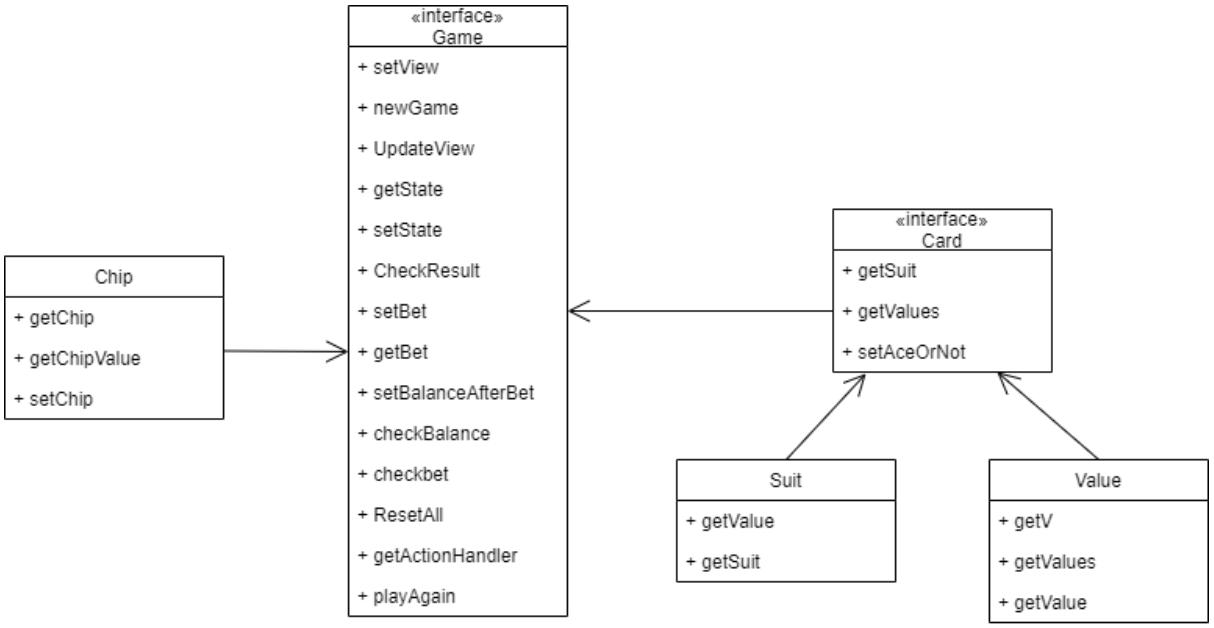

figura 1.1 schema UML dell’analisi del problema

CAPITOLO 2

Design

2.1 Architettura

Per la realizzazione del progetto è stato scelto di implementare il pattern MVC. La suddivisione dell'applicazione con questa architettura permette l'indipendenza necessaria e non rende troppo complicate le possibili modifiche future.

Il Model definisce la logica di dominio dell'applicazione, progettata per modellare i comportamenti delle entità di gioco cioè le carte e le chips.

Il Controller gestisce le interazioni delle entità manipolando i dati del Model comunicando alla View le eventuali modifiche. La classe GameImpl è la classe principale del Controller la quale ha accesso alle classi DealerTurnImp, DealeDrawImp, PlayerTurnImp e PlayerDrawImp.

La View interagisce con il controller tramite ActionHandler che permette di modificare lo stato di tutte le fasi di gioco, per aggiornarla in maniera automatica.

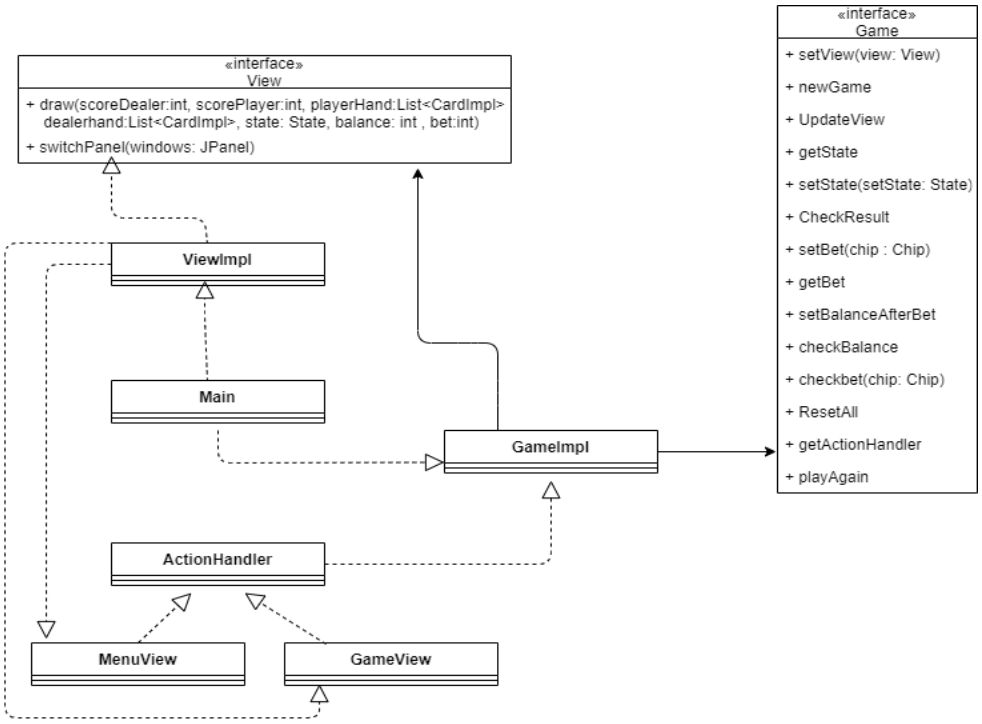

figura 1.2 schema UML architettura

2.2 Design dettagliato

Enrico Baroni

In questa sezione si pone particolare attenzione sull'implementazione della logica del game, la gestione della mano e del turno del Player, la gestione del saldo del Player e infine del model.

Model Card

Il Model delle carte è stato implementato in modo tale da gestire i due enum *Suit e Values* che rappresentano rispettivamente il seme delle carte e i valori delle carte.

Dentro questa classe viene definito il metodo che imposta il valore più conveniente all'asso

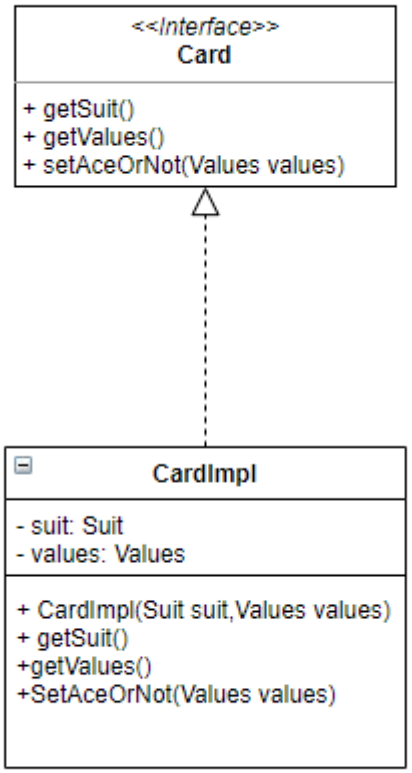

figura 1.3 schema UML della classe CardImpl

Gestione del saldo del Player

Per la gestione del saldo del player ho deciso di creare una variabile nel game. All'avvio del gioco vengono conferiti al player 1000 crediti, ogni volta che viene

effettuata una scommessa viene prima controllata se e valida cioè se il player non sta cercando di scommettere di più di quello che ha, successivamente vengono scalati i soldi scommessi dal saldo e verrà aggiornata la nuova cifra anche nella view.

Quando il saldo sarà “0” si entrerà nello stato di “broke” e il gioco terminerà.

Turno del Player

Nel progetto inoltre mi sono impegnato a sviluppare il turno del player che si avvia appena il giocatore stesso decide quanto scommettere.

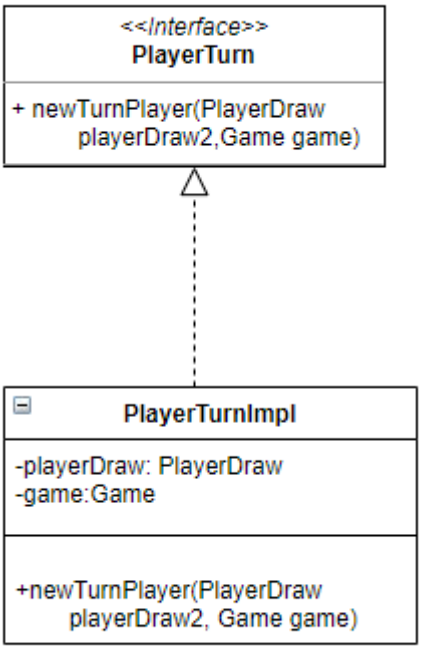

figura 1.4 schema UML della classe PlayerTurnImpl

Creando le classi nel controller PlayerTurn e la sua implementazione PlayerTurnImpl, quando lo stato di gioco è nel turno del Player viene richiamata questa classe che segue le rispettive giocate del player stesso e le esegue, quindi se il player decide di pescare un'altra carta viene richiamata questa classe che al suo interno richiama la classe PlayerDraw che gestice la mano e genera una nuova carta pescata.

Inoltre il playerturn controlla se la mano del player “sballa ” passando così il turno al dealer.

Alberto Rossi

In questa sezione si pone particolare attenzione sull'implementazione del menu principale,del campo da gioco e sulla gestione del turno del dealer.

Menu Principale

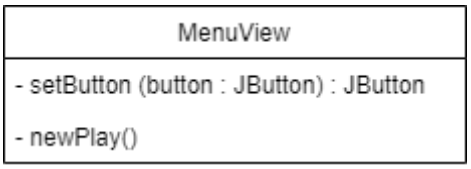

figura 2.1 schema UML della classe MenuView

il menù viene presentato in modo semplice e chiaro, con 3 bottoni diverse che mostrano schermate differenti:

- **Start** : Cambio di scena e visualizza il campo di gioco
- **Credits** : Apertura "showMessageDialog" con informazioni sul progetto
- **Exit** : Chiusura del gioco

Per l'implementazione della grafica del menu e del campo da gioco ho utilizzato la libreria **Java Swing,** creando la finestra principale (view->viewImpl) chiamata "JFrame" a cui tramite un metodo chiamato "switchPanel" inserisco al suo interno i due Jpanel principali, il MenuView e Il GameView.

La classe MenuView(fig 2.1) estende la classe JPanel di swing e al suo interno vengono creati uno a uno i componenti grafici come: i bottoni, il titolo e l'immagine di sfondo. Per la visualizzazione dei componenti in maniera verticale uno sotto l'altro mi sono servito del layout "BoxLayout".

Per settare tutte le specifiche di un determinato bottone ho deciso di creare una classe "setButttton"(fig 2.1) che ritorna il "JButton", questo per evitare ridondanze nel codice.

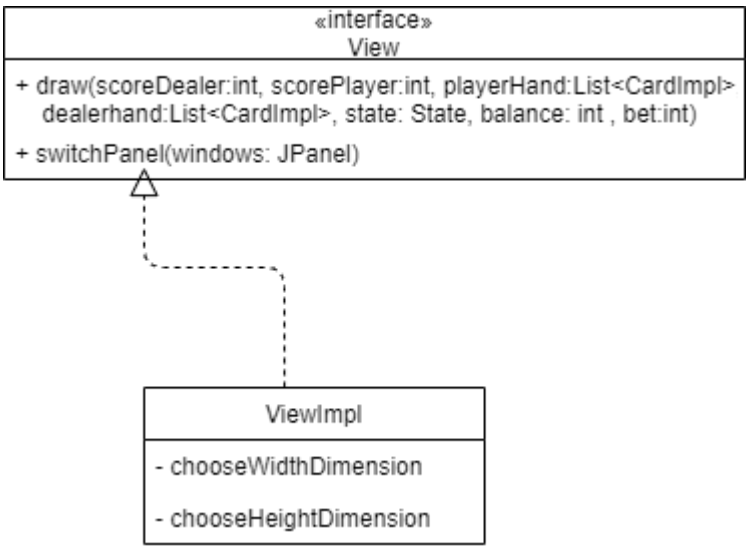

figura 2.2 schema UML della classe view e della sua implementazione

Campo da gioco

Per la creazione del campo da gioco ho optato per la implementazione di ogni componente via codice.

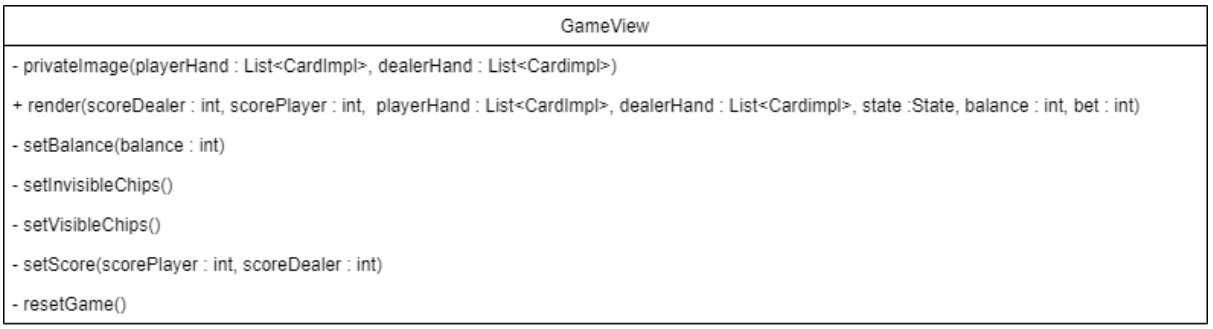

figura 2.3 schema UML della classe gameView

La classe GameView estende la classe Jframe e al suo interno viene creato il campo di gioco che è formato da cinque pannelli:

- **Table :** é il tavolo da gioco dove vengono inseriti i punteggi del dealer e del giocare
- **DealerPanel** : e il “campo” del dealer dove vengono inizializzate 6 JLabel per la visualizzazione delle future carte
- **PlayerPanel** e il “campo” del giocatore dove vengono inizializzate 6 JLabel per la visualizzazione delle future carte

- **ButtonPanel** : è il pannello dove vengono visualizzati i bottoni principale del gioco
- **Chips :** questo pannello si occupa della visualizzazione delle chips di gioco per permetterti di fare la tua puntata ogni inizio gioco

Il GameView muta il suo comportamento in base allo stato di gioco in cui ci troviamo. l'ActionHandler permette di far comunicare la view e il controller, che in base alla situazione richiede l'aggiornamento della view tramite il draw e render nella classe del Game.

Turno del Dealer

Inoltre mi sono occupato della gestione del turno del dealer, che avviene non appena il giocatore o "sballa" o "sta".

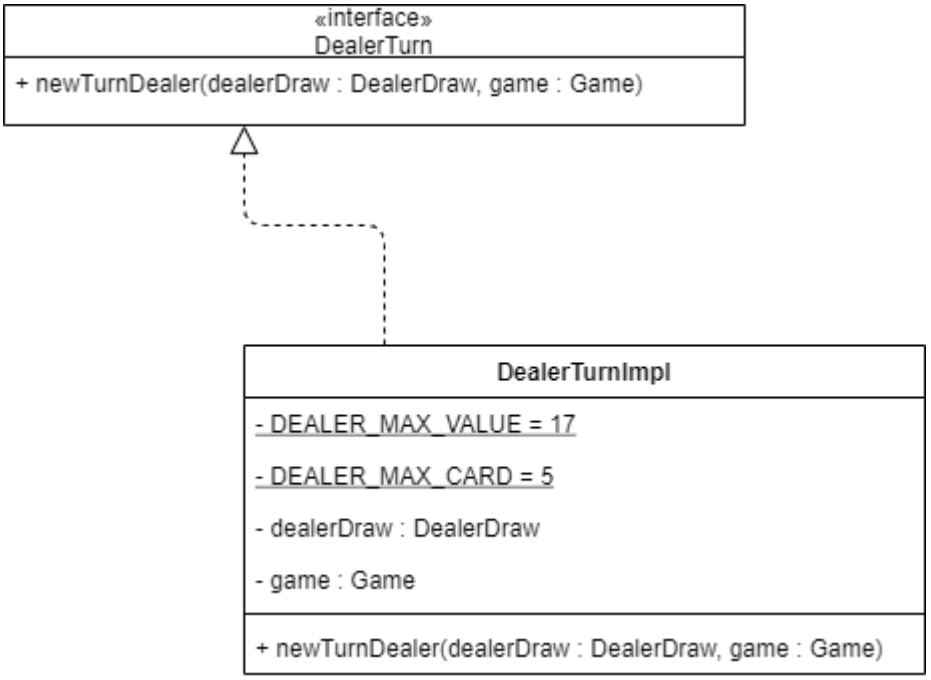

figura 2.4 schema UML della classe DealerTurn

Ho creato due classi nel controller DealerTurn e la sua implementazione DealerTurnImpl, quando lo stato di gioco è nel turno del Dealer viene richiamata questa classe che in base alle regole del "tavolo" pesca le carte utilizzando il metodo implementato dal mio collega dealerDraw. Il dealer dovrà pescare finchè non raggiungerà un punteggio maggiore o uguale a 17 constatando la fine del suo turno, richiamando il chekResult del game controller.

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Per il testing automatizzato è stato utilizzato JUnit 5. La GUI (Graphical User Interface) del menù principale è stata testata manualmente poiché con le conoscenze a disposizione non è stato possibile effettuare test automatizzati in grado di analizzare correttamente la presenza di qualche errore**.**
Per testare le altre specifiche del software, in particolare:

-Enrico Baroni:

Card: Verifica che si assegnino il giusto seme e valore alla carta,e che si cambi correttamente il valore dell'asso
Game: Verifica che ci sia il giusto saldo all'inizio del gioco, che funzionino le scommesse e che siano settati i giusti stati di gioco
PlayerTurn: Verifica che vengano pescate il giusto numero di carte

-Alberto Rossi:

dealerTurn: Verifica che ogni volta che il dealer pesca, venga aggiornata la sua mano
ImageLoader: Verifica che vengano caricate in maniera corretta le immagini di ogni seme

3.2 Metodologia di lavoro

La suddivisione dei ruoli è stata equilibrata e la suddivisione del lavoro è stata rispettata.
Il lavoro è stato diviso principalmente in questo modo:
-**Alberto Rossi**:view, turno del dealer
-**Enrico Baron**i: controller, Turno del player, model

3.3 Note di sviluppo

Alberto Rossi

Io mi sono occupato principalmente della view e del turno del dealer.
Durante l'implementazione della mia parte ho cercato di rendere il mio codice più possibile del codice di qualità, evitando il più possibile ridondanze.
Ho inoltre sfruttato la libreria di java swing per creare al meglio le interfacce di gioco, utilizzando in particolare il Jframe, JPanel,BoxLayout e GridLayout.

Enrico Baroni

Io mi sono occupato della gestione del gioco, degli stati di gioco , del turno del player e infine della parte di model.
Il Game è la classe principale del programma, qui si gestisce l'avvio di una nuova partita, si imposta un nuovo saldo e si danno le prime carte sia al player che al dealer, e infine ha il compito di ripristinare i dati per far partire un nuovo turno oppure una nuova partita.
Gli stati di gioco sono stati gestiti con un enum e una volta entrato in un nuovo stato di gioco si aggiorna la view per impedire alcune azioni al player.

Capitolo 4

Commenti finali

4.1 Autovalutazione e lavori futuri

Alberto Rossi

Secondo il mio punto di vista è venuto fuori un bel progetto, sicuramente si sarebbe potuto fare meglio, ma è stato un lavoro che ci ha aiutato a capire che il lavoro di gruppo aiuta a superare più velocemente qualsiasi scoglio.

In questo periodo ho iniziato a lavorare in una software house e questo lavoro ha aiutato il mio percorso lavorativo, riuscendo meglio a capire le dinamiche del lavoro di gruppo.

Enrico Baroni

Il progetto sotto il mio punto di vista è un'ottima partenza per imparare lo sviluppo di programmi in gruppo.

Da questa esperienza ho imparato che lavorare in coppia è molto importante e soddisfacente, ho capito che ascoltare altre persone ti stimola la conoscenza e diventa più facile risolvere i problemi tramite il confronto.

In questo periodo non è stato facile lavorare insieme perché è mancato il potere vedersi e l’interagire solo tramite computer sicuramente è più limitativo. ma la ritengo in tutto e per tutto una bella esperienza.

Appendice A

Guida utente

La figura A.1 mostra la schermata del menu principale, che presenta tre bottoni:
-**START**: cliccando inizia una nuova partita
-**CREDITS**: cliccando si apre una finestra dove ci sono scritti i contatti (figura A.2)
-**EXIT**:cliccando si chiude l'applicazione

La figura A.3 rappresenta il tavolo di gioco ("prima fase") dove si deve decidere quanti crediti si vuole puntare (se si cerca di puntare di più di quello che si ha si bloccheranno le fish figura A.4), e si cliccherà sul bottone gioca per iniziare una nuova partita.
La figura A.5 rappresenta la partita dove il player decide cosa fare cliccando i due bottoni:
-**PESCA**:si pescherà una nuova carta
-**STAI**:per passare il turno al dealer
Una volta che il turno del dealer è terminato si passerà al resoconto della partita indicando chi ha vinto la partita figura A.6-A.7.

Quando il player finirà i crediti si aprirà una finestra dove ci sarà scritto se si vuole far partire una nuova partita oppure chiudere il gioco figura A.8

figura A.1

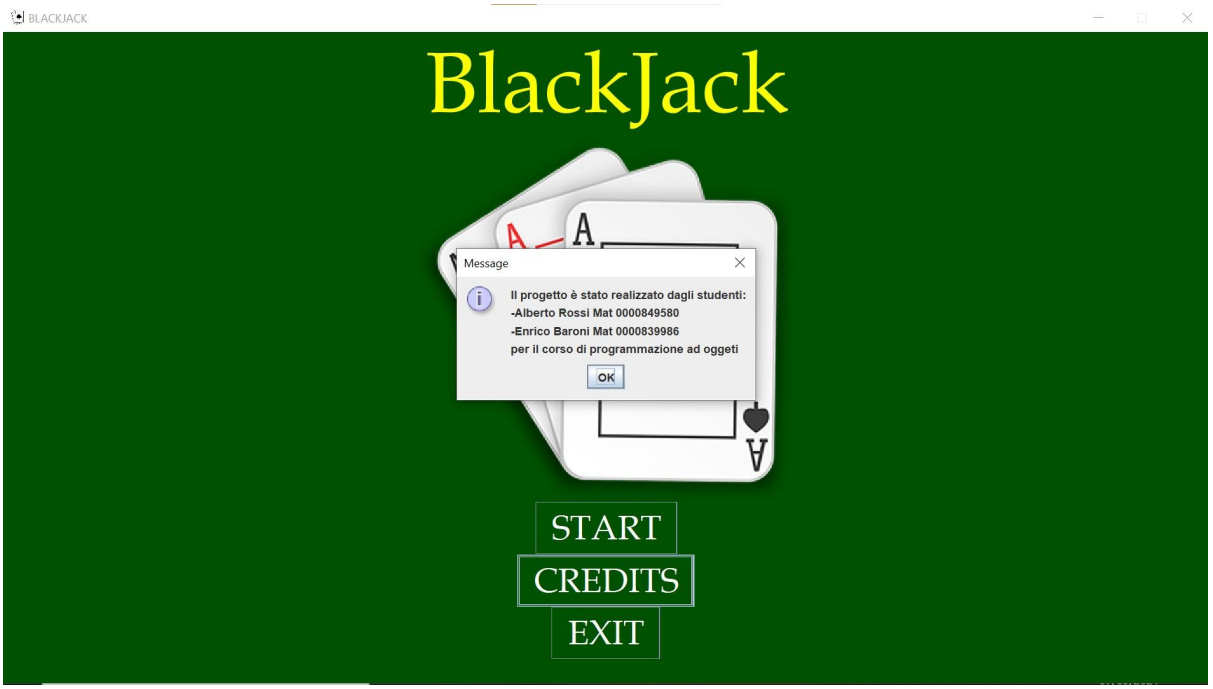

figura A.2

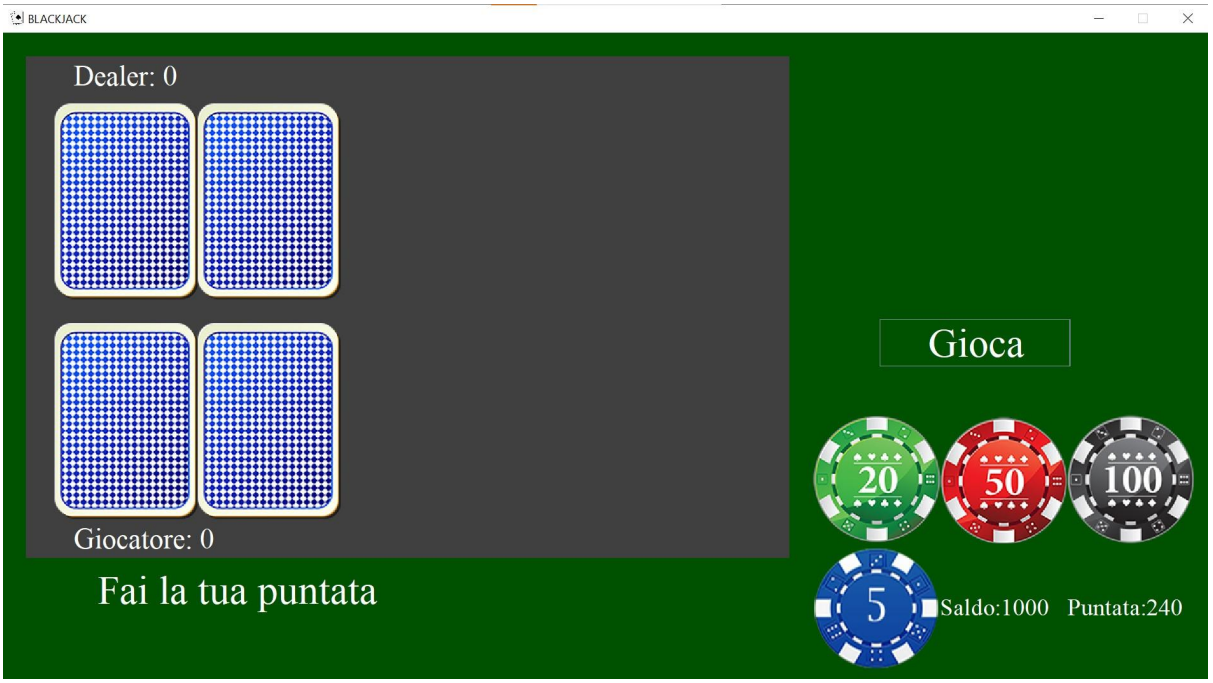

figura A.3

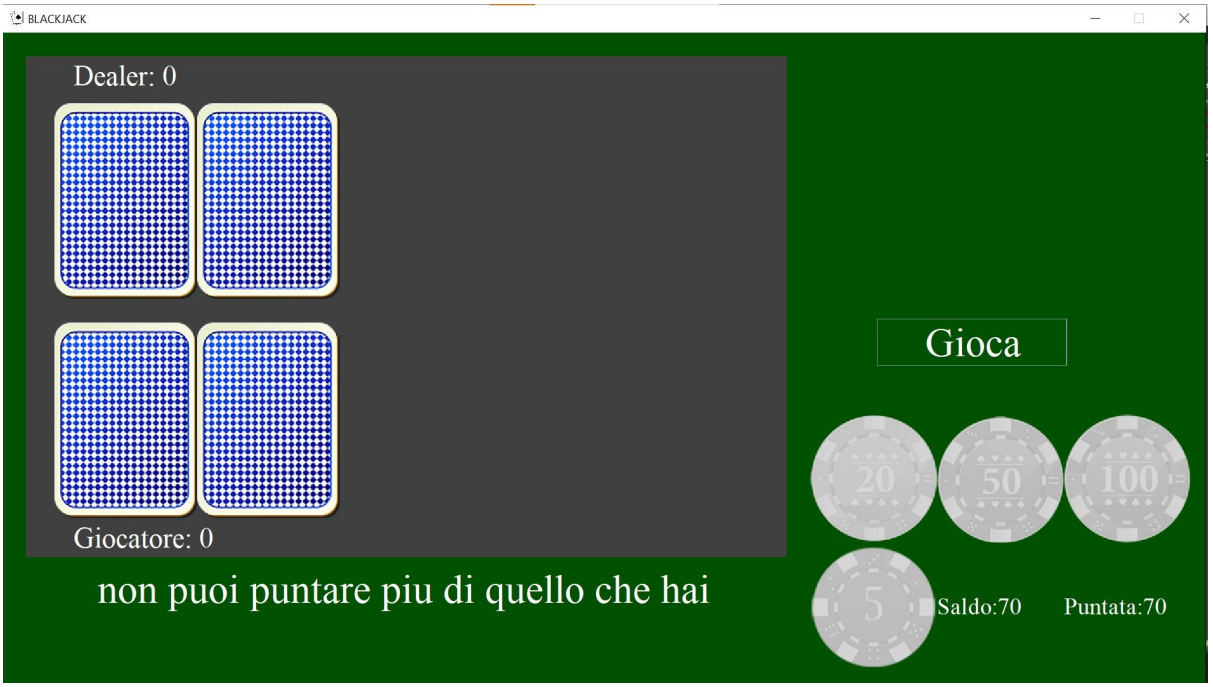

figura A.4

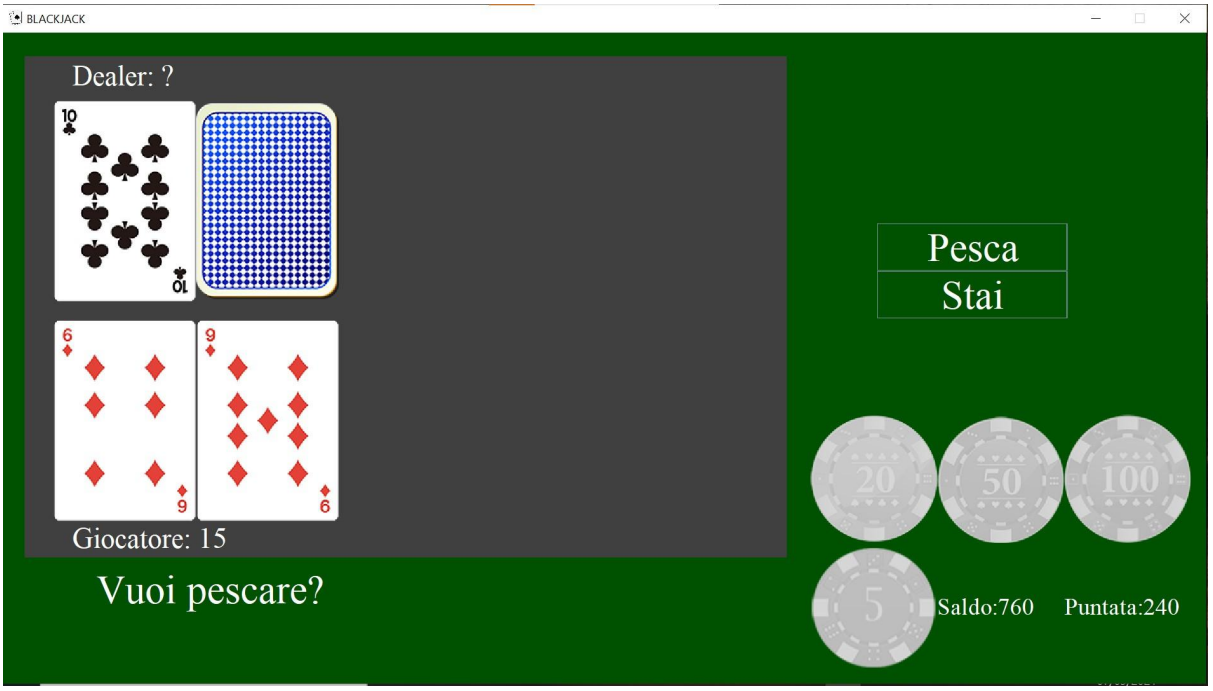

figura A.5

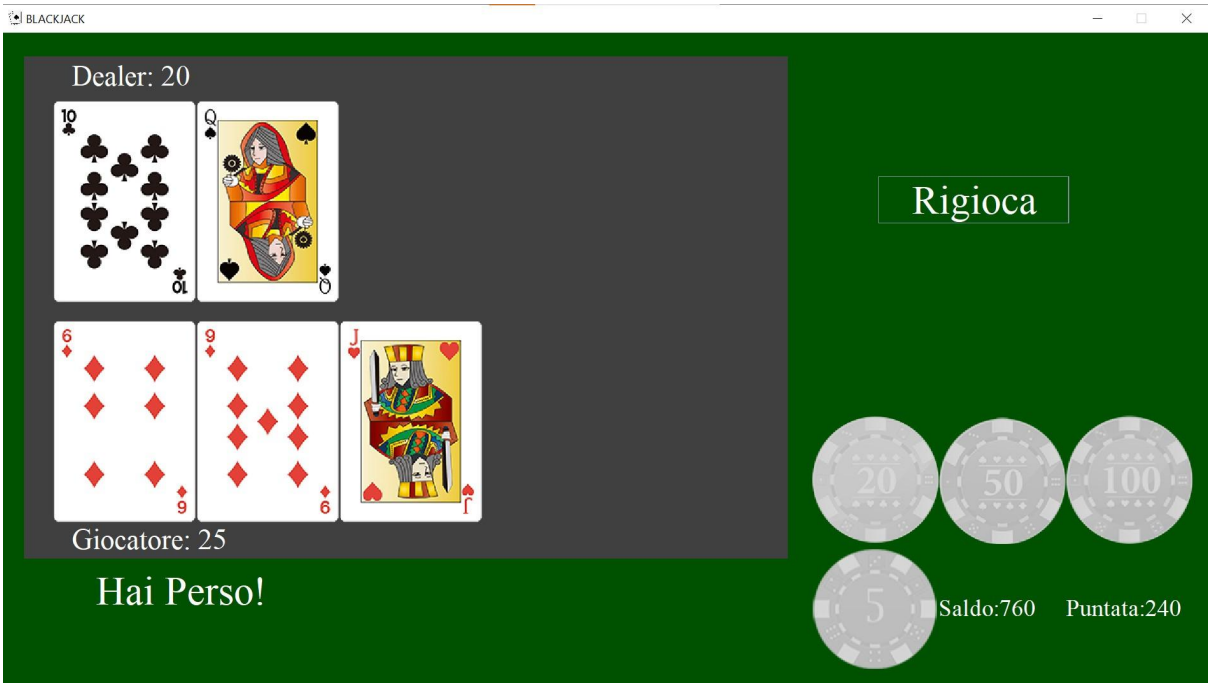

figura A.6

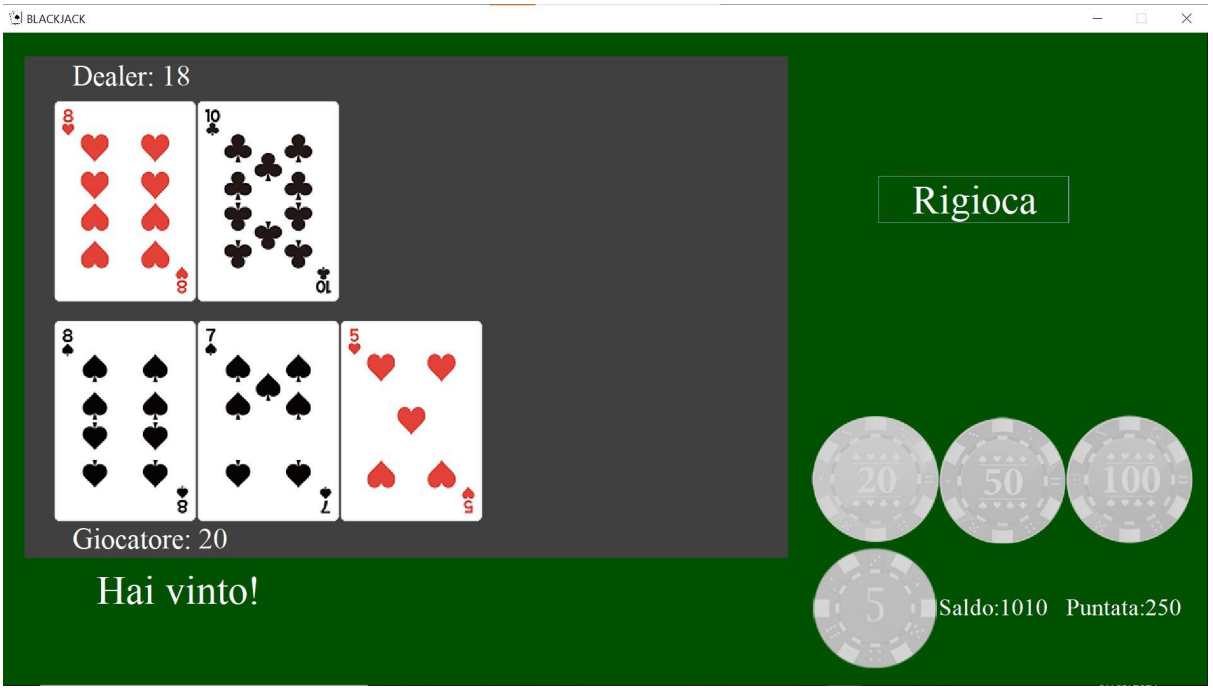

figura A.7

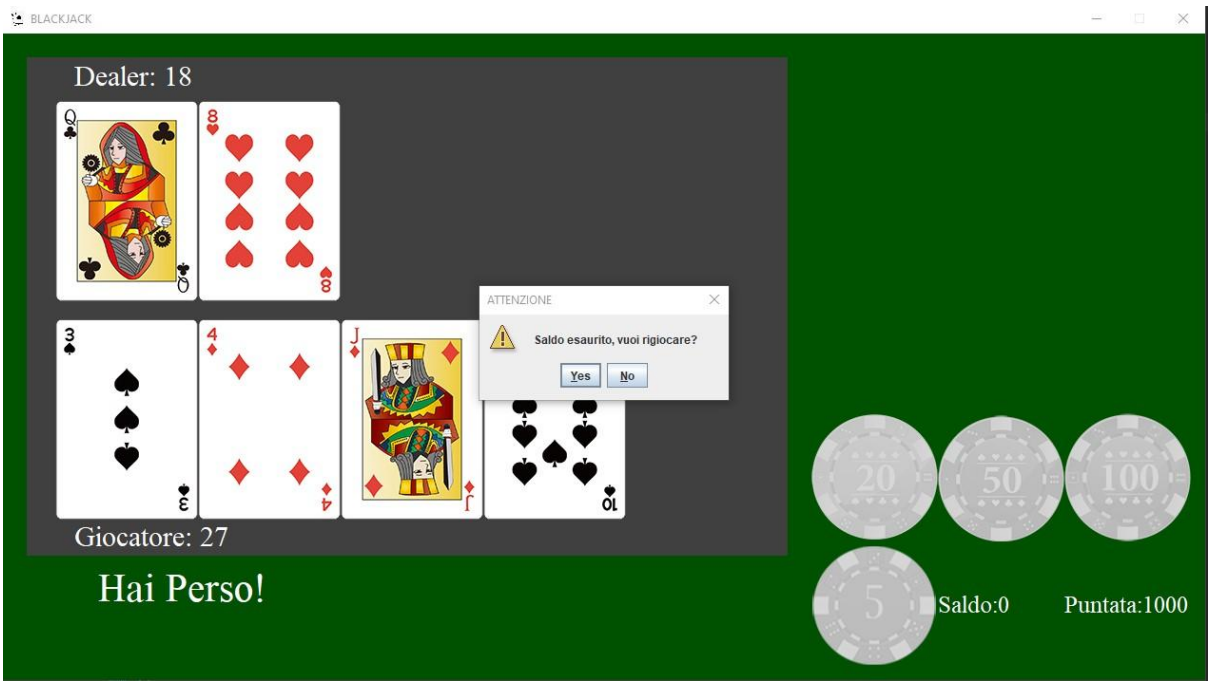

figura A.8

sito per illustrazioni carte e chips

https://chicodeza.com/freeitems/torannpu-illust.html

https://pixabay.com/ja/

https://toppng.com/