

Corso di Progettazione del Software

DOCUMENTAZIONE TECNICA

GIOCO DEL BLACKJACK – MODALITÀ CASINÒ

Studente: Giovanni Ferrentino

Anno accademico: 2024/2025

Indice

1. 1. Introduzione
2. 2. Regole del Blackjack
3. 3. Diagramma UML
4. 4. Gestione Puntate e Saldo
5. 5. Interfaccia Utente
6. 6. Gestione Errori
7. 7. Conclusione

1. Introduzione

Il progetto consiste nello sviluppo di una simulazione del gioco del **Blackjack** in linguaggio **Python**, ispirata alle regole tradizionali adottate nei casinò. L'obiettivo principale è stato quello di applicare i concetti di **programmazione orientata agli oggetti (OOP)**.

2. Regole del Blackjack

Il Blackjack è un gioco di carte, dove l'obiettivo è avvicinarsi a 21 senza superarlo.

Esso si svolge tra un giocatore e il banco (dealer) con un mazzo di 52 carte inglesi senza I jolly.

Introduciamo il valore delle carte:

- Le carte da 2 a 10 valgono il loro numero
- Le figure (J,Q,K) valgono 10
- L'asso A vale 1 o 11, 1 se con l'altro valore (11) supera 21

SVOLGIMENTO GIOCO

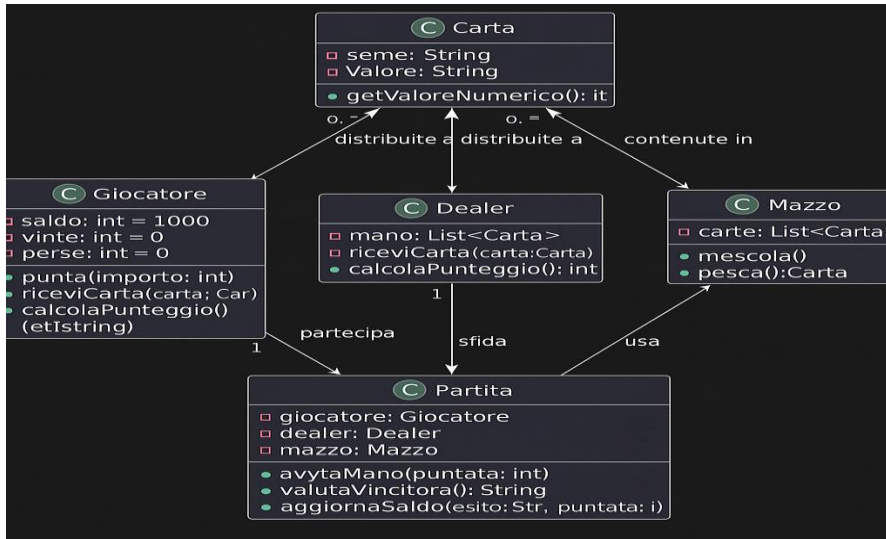
- Il giocatore stabilisce una puntata in monete prima dell'inizio
- Al primo turno, il giocatore riceve due carte scoperte
- Mentre il banco ne riceve due, le quali sono coperte fino a fine partita (ricordiamoci che stiamo parlando del blackjack del casinò)
- Il giocatore dopo aver ricevuto le due carte può decidere se:
 - 1) Chiedere una carta "pesca una carta"
 - 2) Fermarsi "stai"
- Il turno del giocatore termina quando si ferma o quando la somma delle carte supera 21
- Dopo che il turno del giocatore termina, inizia il turno del banco (dealer)
- Le sue carte vengono rilevate a fine partita
- Il banco pesca fino a quando ha ottenuto un punteggio maggiore di 17
- Se il banco supera 21 o totalizza un punteggio più basso del giocatore, perde
- O il giocatore o il dealer se hanno un blackjack naturale (vincono a prescindere se l'altro non ha la stessa mano)
- In base ai risultati, è stata implementata una statistica, dove conta quante vittorie, sconfitte e pareggi sono stati fatti
- Il gioco termina quando le monete sono finite o il giocatore lascia

SISTEMA PUNTATE

- Il giocatore ha un saldo di 1000 monete virtuali
- Se il giocatore perde, perde la quantità di monete che ha puntato a inizio partita
- Se pareggia riprende la scommessa

- Se vince , si raddoppia la sua scommessa iniziale

3. Diagramma UML



Parliamo di ogni entità (ogni oggetto è una classe)

L'oggetto Carta ha:

- seme (es. cuori, fiori)
- valore (es. 10, Jack, Asso)
- Il metodo `getValoreNumerico()` serve per convertire una carta in un punteggio numerico, utile per calcolare il totale della mano.

L'oggetto Giocatore presenta vari metodi

- `Punta ()`, che importa l'importo da scommettere
- `riceviCarta` e ho messo l'oggetto carta per far sic he il giocatore riceva la carta
- `calcola punteggio ()` somma I valori delle carte
- `calcola punteggio()` implementato per contare quante partite sono state perse,pareggiate e vinte

L'oggetto dealer (implementato come una classe a parte):

- lo separo da giocatore perchè esso non deve puntare
- alla fine ha un comportamento automatico poichè lui deve prendere la carta fino a quando non supera 17, ovviamente cercando di non superare 21
- per metterlo in relazione con giocatore, è stato implementato la classe partita

Per quanto riguarda Mazza, che anche esso è stato implementato a parte per vari motivi:

- per rendere il codice molto piu chiaro e leggibile
- incapsula tutte le cose che possiamo fare con le carte (creazione ecc..)
- infatti Mazza prende la lista di carte e crea un mazzo

Passiamo a quella che è la classe cruciale, ossia, Partita:

- gestisce il gioco mettendo in relazione dealer,giocatore e mazzo e tiene le loro istanze
- Infatti valuta mano del giocatore e dealer
- Ha una logica per decider il vincitore o meno
- Aggiorna il saldo del giocatore

4. Gestione Puntate e Saldo :

- il giocatore parte con un saldo di 1000 monete
 - il saldo presenta quanto un giocatore può scommettere
 - prima di ogni mano il giocatore decide quando giocare
 - ovviamente importo positivo e minore uguale al saldo (è stato implementato in modo che sia così)
 - dopo la fine della partita ci sono 3 situazioni:
 - 1) se il giocatore perde la partita , perde anche la puntata
 - 2) se il giocatore pareggia, riceve la puntata
 - 3) se il giocatore vince, riceve il doppio della puntata
- Ecco perchè abbiamo aggiornato saldo alla classe puntata
- Il gioco termina in due casi:
- Il giocatore esce volontariamente
 - Il suo saldo scende a zero

5. Interfaccia Utente

È stata successivamente integrata una **Web App interattiva** utilizzando **Streamlit**, per migliorare l'esperienza utente:

1) Controlli grafici:

- Pulsanti PESCA (Hit) e STAI (Stand)
- Inserimento puntata tramite campo numerico
- Pulsante Nuova Mano per ripartire

2) Visualizzazione dinamica:

- Le carte vengono mostrate con nomi e simboli
- Il saldo viene aggiornato a ogni mano
- Messaggi informativi chiari in tempo reale

3) Accessibilità:

- L'interfaccia è fruibile da browser
- Può essere condivisa online con un link pubblico

<https://blackjack-cfupvbzvmisz7hqjdlx7zx2.streamlit.app> (link app)

6. Gestione Errori

Il mazzo viene ricreato e mescolato ad ogni nuova mano, quindi non si presentano problemi legati all'esaurimento delle carte. Tuttavia, l'uso della classe Mazzo incapsula questa logica per sicurezza e riusabilità.

Nella versione Streamlit:

-Tutti gli input sono **controllati da widget numerici e pulsanti**, quindi l'utente **non può inserire dati errati**.

-Le eccezioni (es. saldo insufficiente) sono gestite con messaggi `st.warning()` o `st.error()`.

7. Conclusione

Grazie a questo progetto, abbiamo messo in pratica le nostre conoscenze su come progettare e organizzare un sistema software utilizzando classi, metodi e interazioni tra oggetti, seguendo i principi della progettazione modulare.

La suddivisione in classi ben definite (Carta, Mazzo, Giocatore, Dealer, Partita) ha consentito di modellare in maniera efficace la logica e le dinamiche del gioco reale, mantenendo il codice chiaro, manutenibile ed estendibile.

Il progetto ha permesso di:

- Applicare in modo concreto i concetti teorici del corso
- Costruire una base solida per eventuali estensioni future (web app, multiplayer, ecc)
- Farci un'idea di quello che è il processo creativo di qualunque piattaforma usiamo (social, giochi ecc) ma ovviamente con dovute differenze
- Usare un approccio critico per far sì che il gioco funzioni anche su Stramlit

La realizzazione dell'interfaccia web tramite **Streamlit** ha inoltre migliorato l'usabilità del gioco, consentendo un'interazione intuitiva e moderna, facilmente accessibile anche a chi non ha conoscenze tecniche.

