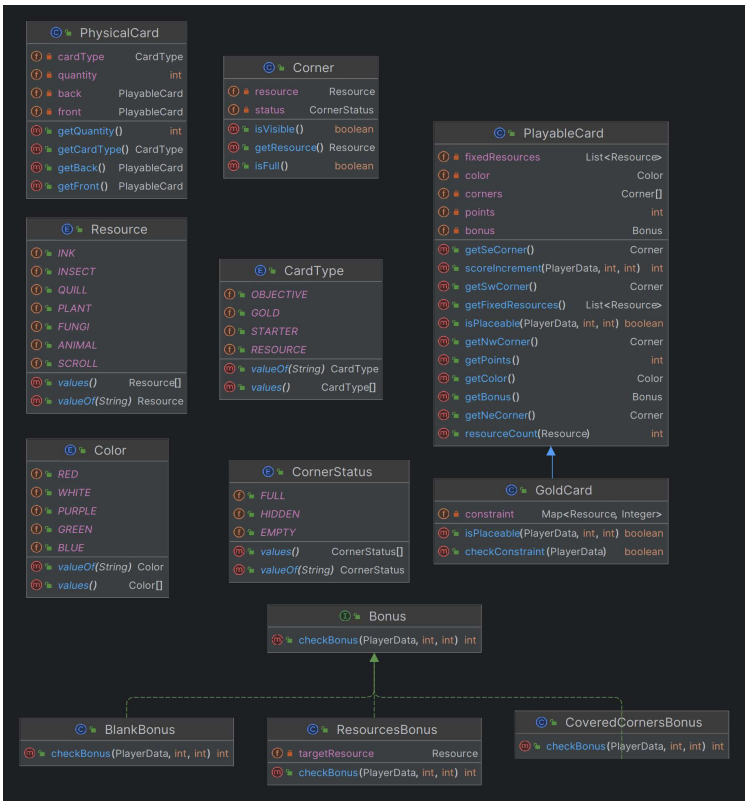
DESCRIZIONE UML PROGETTO GC50:

Nel file UML le classi sono state divise in quattro package:

CARD: rappresentano le carte



La classe **PhysicalCard** rappresenta la carta in senso di oggetto fisico con un fronte e un retro, sono presenti i getter per accedere agli attributi.
La classe **PlayableCard** rappresenta le carte come l'insieme delle proprietà e dei valori che esse assumono durante il gioco. Gli attributi sono il colore, i 4 angoli, i punti gli eventuali bonus e le risorse indicate. Ciascuna di questi elementi è stata espressa mediante una classe enumerazione, dove sono indicati i valori, ad esempio per la classe **Color** ha tutti i colori delle carte e i metodi che restituiscono i colori.
La classe **GoldCard** è una sottoclasse di PlayableCard e ne eredita per questo tutti gli attributi, inoltre definisce anche il vincolo chiamato constraints che indica quante risorse bisogna possedere per poterla utilizzare.
L'enumerazione **Resource** indica il tipo di risorsa a cui la carta appartiene, mentre **CardType** indica se la carta è OBIETTIVO, GOLD, RISORSA O INIZIALE.
Gli angoli sono stati indicati in corrispondenza dei punti cardinali sw,ne,nw,se, definiti nella classe **Corner**.
Un angolo può essere coperto, vuoto o contenente una risorsa come si nota dall'enumerazione **CornerStatus.**
La classe **Bonus** indica i punti aggiuntivi che vengono dati, le sue sottoclassi sono **BlankBonus**, **ResourceBonus** e **CoveredCornerBonus**.
L'implementazione delle carte permette di aggiungere nuove tipologie, rendendo così il programma flessibile.

GAME:

Le carte sono salvate all'interno di una matrice, la classe **CardMatrix** contiene tutti i metodi necessari per gestire la matrice quali l'inserimento, la visualizzazione e l'aggiunta delle carte all'interno di essa.

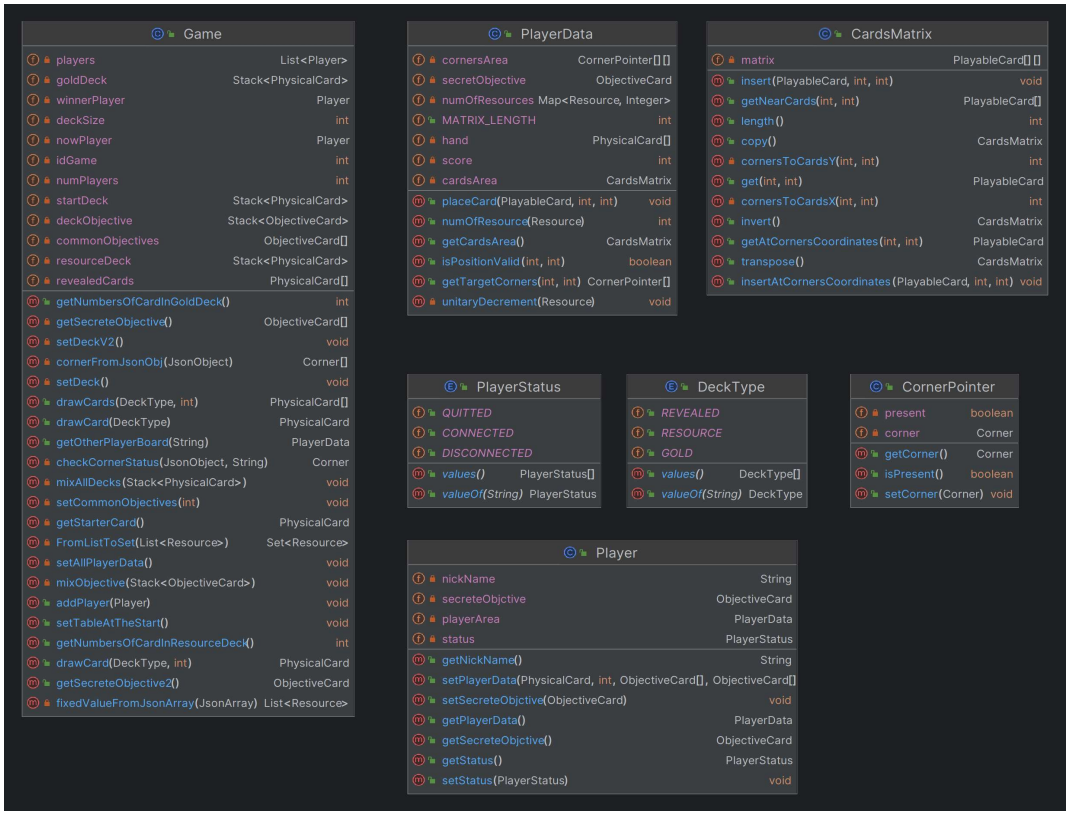
PlayerData contiene tutti i dati del giocatore, relativi alla partita che sta giocando, quali carte che ha in mano, le carte obiettivo segreto, il punteggio e così via. I suoi metodi permettono di visualizzare se la posizione di una carta è valida oppure il numero di risorse possedute.

Del giocatore vengono memorizzate anche i dati nella classe **Player**, tra cui lo status memorizzato nell'enumerazione **PlayerStatus** i cui possibili valori sono CONNECTED,QUITTED, DISCONNECTED.

La classe **CornerPointe**r indica qual è l'angolo in cui ci si trova, mentre l'enumerazione **DeckType** indica se la carta è RESOURCE, REVEALED o GOLD.

La classe **Game** infine contiene i dati relativi all'intera partita, dunque non solo quelli del singolo giocatore ma anche quelli degli altri giocatori ad esempio il vincitore, l'elenco dei giocatori e la pile di carte Gold e risorse da cui è possibile pescare.

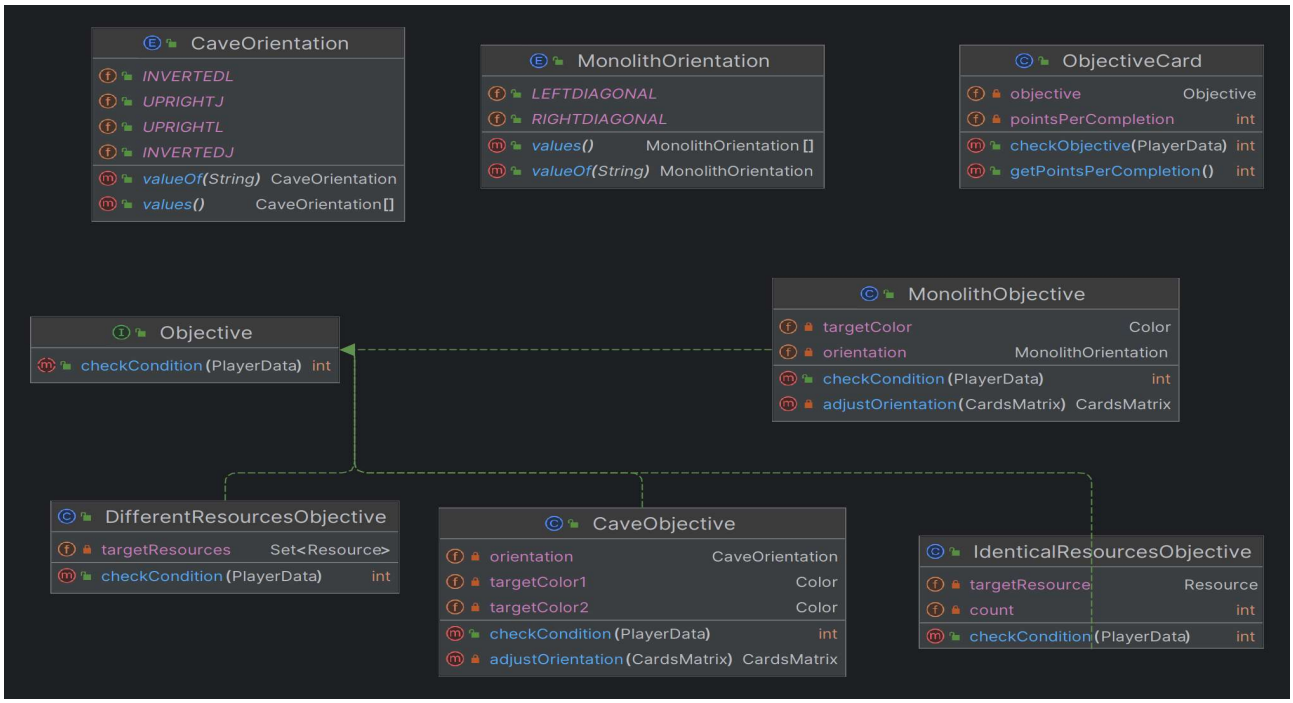
La matrice delle carte servirà anche per visualizzare l'area di gioco mentre quella dei corner servirà per supportare la logica relativa al posizionamento delle carte e il numero di risorse in possesso dei giocatori.



OBJECTIVE: rappresenta le carte obiettivo
La classe Objective ha 4 sottoclassi:

- **MonolithObjective** che rappresenta gli obiettivi delle carte con il monolite
- **CaveObjective** che rappresenta gli obiettivi delle carte con la caverna
- **IdenticalResourcesObjective** che rappresenta gli obiettivi per le risorse uguali
- **DifferentResourcesObjective** che rappresenta gli obiettivi delle carte con differenti risorse.

ObjectiveCard rappresenta le carte obiettivo e i suoi attributi indicano il tipo di obiettivo e i punti per raggiungerlo. Le enumerazioni **MonolithOrientation** e **CaveOrientation** descrivono le possibili disposizioni delle carte necessarie per raggiungere l'obiettivo.



CHAT:

La chat è implementata tramite due classi: **Chat** che presenta i metodi addMessage e getMessage che permettono di aggiungere o visualizzare un messaggio, e la classe **Message** che implementa il messaggio scambiato i suoi attributi indicano l'ora, il mittente, il destinatario e il contenuto. I metodi getter permettono di visualizzare i rispettivi attributi.