

## 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 **CornerPointer** 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.

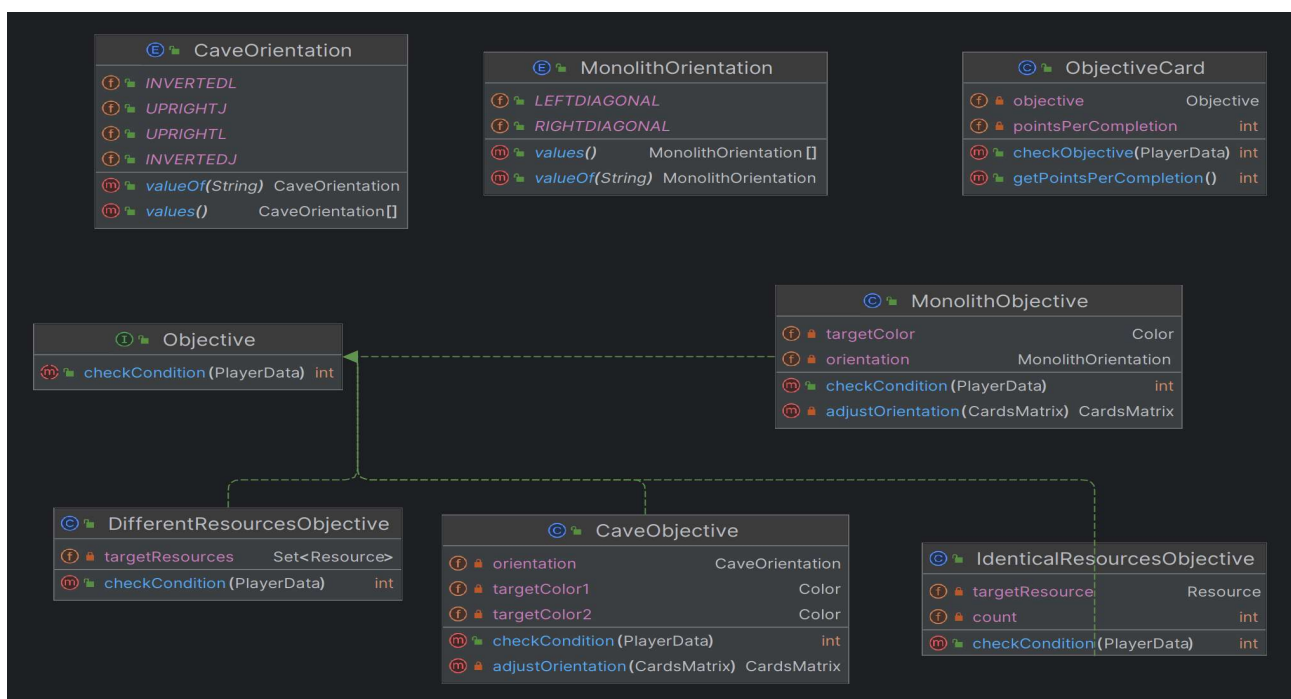
<div><div>Game</div><div><div><div>players</div><div>List&lt;Player&gt;</div></div><div><div>goldDeck</div><div>Stack&lt;PhysicalCard&gt;</div></div><div><div>winnerPlayer</div><div>Player</div></div><div><div>deckSize</div><div>int</div></div><div><div>nowPlayer</div><div>Player</div></div><div><div>idGame</div><div>int</div></div><div><div>numPlayers</div><div>int</div></div><div><div>startDeck</div><div>Stack&lt;PhysicalCard&gt;</div></div><div><div>deckObjective</div><div>Stack&lt;ObjectiveCard&gt;</div></div><div><div>commonObjectives</div><div>ObjectiveCard[]</div></div><div><div>resourceDeck</div><div>Stack&lt;PhysicalCard&gt;</div></div><div><div>revealedCards</div><div>PhysicalCard[]</div></div><div><div>getNumbersOfCardInGoldDeck()</div><div>int</div></div><div><div>getSecreteObjective()</div><div>ObjectiveCard[]</div></div><div><div>setDeckV2()</div><div>void</div></div><div><div>cornerFromJsonObj(JsonObject)</div><div>Corner[]</div></div><div><div>setDeck()</div><div>void</div></div><div><div>drawCards(DeckType, int)</div><div>PhysicalCard[]</div></div><div><div>drawCard(DeckType)</div><div>PhysicalCard</div></div><div><div>getOtherPlayerBoard(String)</div><div>PlayerData</div></div><div><div>checkCornerStatus(JsonObject, String)</div><div>Corner</div></div><div><div>mixAllDecks(Stack&lt;PhysicalCard&gt;)</div><div>void</div></div><div><div>setCommonObjectives(int)</div><div>void</div></div><div><div>getStarterCard()</div><div>PhysicalCard</div></div><div><div>FromListToSet(List&lt;Resource&gt;)</div><div>Set&lt;Resource&gt;</div></div><div><div>setAllPlayerData()</div><div>void</div></div><div><div>mixObjective(Stack&lt;ObjectiveCard&gt;)</div><div>void</div></div><div><div>addPlayer(Player)</div><div>void</div></div><div><div>setTableAtTheStart()</div><div>void</div></div><div><div>getNumbersOfCardInResourceDeck()</div><div>int</div></div><div><div>drawCard(DeckType, int)</div><div>PhysicalCard</div></div><div><div>getSecreteObjective2()</div><div>ObjectiveCard</div></div><div><div>fixedValueFromJsonArray(JsonArray)</div><div>List&lt;Resource&gt;</div></div></div></div>	<div><div>PlayerData</div><div><div><div>cornersArea</div><div>CornerPointer[]</div></div><div><div>secretObjective</div><div>ObjectiveCard</div></div><div><div>numOfResources</div><div>Map&lt;Resource, Integer&gt;</div></div><div><div>MATRIX_LENGTH</div><div>int</div></div><div><div>hand</div><div>PhysicalCard[]</div></div><div><div>score</div><div>int</div></div><div><div>cardsArea</div><div>CardsMatrix</div></div><div><div>placeCard(PlayableCard, int, int)</div><div>void</div></div><div><div>numOfResource(Resource)</div><div>int</div></div><div><div>getCardsArea()</div><div>CardsMatrix</div></div><div><div>isPositionValid(int, int)</div><div>boolean</div></div><div><div>getTargetCorners(int, int)</div><div>CornerPointer[]</div></div><div><div>unitaryDecrement(Resource)</div><div>void</div></div></div></div>	<div><div>CardsMatrix</div><div><div><div>matrix</div><div>PlayableCard[] []</div></div><div><div>insert(PlayableCard, int, int)</div><div>void</div></div><div><div>getNearCards(int, int)</div><div>PlayableCard[]</div></div><div><div>length()</div><div>int</div></div><div><div>copy()</div><div>CardsMatrix</div></div><div><div>cornersToCardsY(int, int)</div><div>int</div></div><div><div>get(int, int)</div><div>PlayableCard</div></div><div><div>cornersToCardsX(int, int)</div><div>int</div></div><div><div>invert()</div><div>CardsMatrix</div></div><div><div>getAtCornersCoordinates(int, int)</div><div>PlayableCard</div></div><div><div>transpose()</div><div>CardsMatrix</div></div><div><div>insertAtCornersCoordinates(PlayableCard, int, int)</div><div>void</div></div></div></div>	<div><div>PlayerStatus</div><div><div><div>QUITTED</div></div><div><div>CONNECTED</div></div><div><div>DISCONNECTED</div></div><div><div>values()</div><div>PlayerStatus[]</div></div><div><div>valueOf(String)</div><div>PlayerStatus</div></div></div></div>	<div><div>DeckType</div><div><div><div>REVEALED</div></div><div><div>RESOURCE</div></div><div><div>GOLD</div></div><div><div>values()</div><div>DeckType[]</div></div><div><div>valueOf(String)</div><div>DeckType</div></div></div></div>	<div><div>CornerPointer</div><div><div><div>present</div><div>boolean</div></div><div><div>corner</div><div>Corner</div></div><div><div>getCorner()</div><div>Corner</div></div><div><div>isPresent()</div><div>boolean</div></div><div><div>setCorner(Corner)</div><div>void</div></div></div></div>	<div><div>Player</div><div><div><div>nickName</div><div>String</div></div><div><div>secreteObjective</div><div>ObjectiveCard</div></div><div><div>playerArea</div><div>PlayerData</div></div><div><div>status</div><div>PlayerStatus</div></div><div><div>getNickName()</div><div>String</div></div><div><div>setPlayerData(PhysicalCard, int, ObjectiveCard[], ObjectiveCard[])</div><div>void</div></div><div><div>setSecreteObjective(ObjectiveCard)</div><div>void</div></div><div><div>getPlayerData()</div><div>PlayerData</div></div><div><div>getSecreteObjective()</div><div>ObjectiveCard</div></div><div><div>getStatus()</div><div>PlayerStatus</div></div><div><div>setStatus(PlayerStatus)</div><div>void</div></div></div></div>
--	---	--	--	--	---	---

**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.

