\*\*le classi possono avere un nome diverso da quello dell’uml

* Il Server viene acceso, e apre 3 nuovi threads NewGameManager(Integer nMates) bloccanti la coda di attesa specifica, con ConcurrencyManager. Questi threads rimangono dormienti (liberando la coda) fino a che StartGame, vedi più in basso, chiama notifyAll sulla coda specifica. Questo sistema distribuito alleggerisce il carico sul server, e permette comunque di gestire più partite contemporaneamente.
* Il Client preme il tasto start, accede alla schermata principale del gioco, seleziona il numero di giocatori e preme StartGame. In questa prima versione non è permesso che il giocatore esca durante la fase di connessione, e non è permesso il gioco singolo. Void StartGame(this, nMates), funzione sul server, sincronizzata sulla coda specifica, raggiuge il server che salva il riferimento al client nella coda appropriata, chiama notifyAll e ritorna, chiudendo il thread aperto.
* La chiamata a notifyAll viene recepita dal thread NewGameManager appropriato, e se è il caso apre con ConcurrencyManager il nuovo thread GameManager(ArrayList Players, Integer nMates) che si occuperà di tutto il resto del gioco assieme alla classe Game, creata da TurnManager alla sua creazione, elimina i clients dalla coda. Attenzione a Players, non sono classi dei giocatori virtuali, ma sono i riferimenti ai client tali per cui si possa fare RMI, ad es Players.get(i).updateMatch(), con updateMatch() funzione sul Client. tryStartMatch ricomincia il while e si addormenta di nuovo, giustamente 😊.
* La prima cosa che GameManager fa è aggiornare i clients che l’accordo è stato raggiunto, inviando le possibili carte/ frame da cui scegliere a ciascun client. Questa operazione deve essere fatta dal server, in questo modo si esclude la possibilità che i client scelgano le stesse carte. La grafica qui ha libero arbitrio se far vedere al giocatore solo tali carte o meno.

Una volta finita questa operazione locale sui client, essi invieranno le loro classi Match e Player con le variabili di stato aggiornate. Il server farà quindi un merge di queste informazioni e restituirà ai client la versione corretta con cui aggiornare la grafica, ovvero le carte pubbliche, i segnalini e tutto il resto.

* Inizia ora la gestione dei turni, la grafica da questo momento dovrà rappresentare il gioco in modo che ogni giocatore veda anche lo stato degli altri giocatori.
* Il player scelto a caso dal server inizia il turno. Questo è reso possibile dalla funzione listen sul server che attiva i pulsanti di un giocatore chiamando una funzione sul client tramite RMI, e mettendo i pulsanti degli altri in grigio. Per eviteare problemi dovuti alla non affidabilità della rete internet, deve essere fatta una verifica anche sul server che permetta solo al client selezionato di effettuare chiamate, scartando le altre. Una prima ipotesi, se anche la parte di network è d’accordo, potrebbe essere un controllo della variabile che rappresenta il client del giocatore attivo, sul server, e quella inviata come parametro alla funzione che è stata chiamata “in ritardo”.
* Al primo turno delle frecce indicheranno al giocatore quali tasti premere, e in locale la logica si occuperà di gestire ed effettuare operazioni corrette. Potrà quindi trare un dado, attivare una carta, uscire, parte o nessuna di queste cose, ma obbligatoriamente dovrà premere completa il turno. Ad ognuno di queste opzioni corrisponderà un tasto. La void quit(this), funzione sul server, chiamata dal client, porrà il client in questione nella variabile toQuit, e poi chiamerà automaticamente la endTurn(this), funzione sul server. Il turnManager, vedendo la fine del turno passa alla riga successiva, controlla se la variabile toQuit è assegnata a un giocatore e in tal caso comunica le informazioni del caso ai giocatori e chiude il thread GameManager con un return . Ovviamente il tutto dovrà essere sincronizzato dalla parte di logica.
* Si procede dunque così fino a quando i turni finiscono, e il server chiama score(), funzione sui client, che farà il calcolo del proprio punteggio e lo invierà al server. La grafica qui puù mostrare quello che preferisce fino all’invio dei risultati da parte del server. Al vincitore sarà mostrata una animazione differente.
* La versione attuale del gioco non prevede una clasifica generale, tantomeno ua chat di gruppo. Le aggiunte obbligatorie (giocatore singolo...) saranno implementate per ultime, quelle opzionali (chat di gruppo...) previa situazione positiva per gli esami.