DESCRIZIONE UML CONTROLLER

MEMBRI DEL GRUPPO

- Polizzi Chiara
- Raimondi Francesco
- Rubagotti Andrea
- Santagata Maria Concetta (REFERENTE)

FUNZIONALITÀ AGGIUNTIVE

Sono presenti classi anche per le 2 funzionalità aggiuntive che abbiamo deciso di implementare (chat, partite multiple).

DESCRIZIONE UML

N.B: L'UML è di alto livello, di conseguenza è privo di tutti i metodi getter e setter di ogni classe in esso presente (li consideriamo sottintesi).

Sono presenti due classi:

- **ServerController**: si occupa delle azioni pre-accesso ad una partita. Tiene tutti i GameController all'interno di una Map (la cui chiave di ricerca è l'id del GameController, generato tramite un attributo di classe firstAvailableId). L'attributo firstAvailableId serve per settare gli Id dei GameControllers che vengono creati: GameController ed il relativo Game avranno il medesimo id. La List allNicknames tiene memoria dei nickname già selezionati (da qui i metodi necessari isNicknameAvailable() e chooseNickname()). Gli altri 2 metodi sono necessari a fornire ad ogni Client la scelta di "creare una partita" o di "aggiungersi ad una partita a scelta già esistente". E' presente anche una List che conterrà i riferimenti ai Client "in lobby"(TCP e RMI): crediamo sia utile al fine di avere i riferimenti anche per gli Observer.
- **GameController**: ad ogni Game (ogni partita) è associato un GameController. Contiene tutti i metodi necessari ai Client per effettuare tutte le azioni all'interno di una partita. Gli attributi lastRounds e lastDrawingRounds servono per gestire gli ultimi turni della partita una volta attivata la situazione di terminazione (20 punti raggiunti o entrambi i mazzi di pesca finiti). La funzione leaveGame() verrà chiamata nel momento in cui un player si disconnette dalla partita (volontariamente o meno). La funzione nextPhase() serve per far procedere i vari turni. La funzione sendMessage() permette ai giocatori di inviare un messaggio (chat a coppia o chat generale). La List winners contiene il vincitore o i giocatori che hanno pareggiato. Implementa l'interfaccia Remote perchè così facendo riusciamo a consegnare ai ClientRMI un oggetto remoto per poter dialogare direttamente senza intasare la connessione. Contiene anche lui una List con i riferimenti ai Client (TCP e RMI) per poterli aggiungere come Observers alle classi del model.