InGame Communication

Messages from Server to Client

- **0.** sending MAX Players: "maxPlayer": int
- **2.** sending every player when InGameActivity is created
    - **2.1** "initialMe" : Player Object
    - **2.2** "initialOtherPlayer": Object
- **4.** after receiving "memorizedCards" send "nextPlayer": PlayerObject

- **6.** after receiving "pickCard" send "PickedCard": Card Object
- **8.** after receiving "swapPickedCardWithOwnCards" send to all clients : "discardedCard": Card Object (dies ist die Karte, die nach dem Tausch auf dem Ablagestapel liegt) (Pauline: Symbol und Karte auf Ablagestapel anzeigen)
- **8.** after receiving "playPickedCard" send to all clients : "playedCard": Card Object (Pauline: nur karte auf Ablagestapel anzeigen)
- **9.** after receiving "swapPickedCardWithOwnCards" or "playPickedCard" send to all player the Player Object, who's on the turn -> "updatePlayer": Player Object

When player played a card with a functionality:
- **11.** after receiving " useFunctionalityPeek" sending to all " useFunctionalityPeek": Card Object
- **11.** after receiving "useFunctionalitySpy" sending to all " useFunctionalitySpy": Card Object , Player Object
- **11.** after receiving "useFunctionalitySwap": card Object1, Player1 Object, card Object2, Player2 Object + sending "updatePlayer" for the 2 players which have swaped cards

- **13.** after receiving "finishMove": repeat step 4 and so on
- **15.** send "calculatedScores" after cabo was called: "sendScore": Player Object

Messages from Client to Server
- **1.** sending "askForInitialSetting": this makes sure that whole activity is built before something is sent by the server
- **3.** approving that initial cards have been looked at send "memorizedCards": boolean

Messages from Client when it is his turn:
- **5.** pick card from draw pile : "pickCard": String

after receiving "pickedCard" :

- **7.**swap picked Card with own card : "swapPickedCardWithOwnCards": Card Object
- **7.**play Card (use functionality): "playPickedCard": String
  - **10.**" useFunctionalityPeek": Card Object
  - **10.**"useFunctionalitySpy": cardObject, Player Object
  - **10.**"useFunctionalitySwap": card Object1, Player1 Object, card Object2, Player2 Object,

- **12.** after the player finished the move: send "finishMove": String
- **14.**when a client clicks cabo: send "cabo": String

Note : For steps 7, 8, 10 and 11, only one of these steps can be performed at the same time

Note: Some Messages like "sendMaxPoints"or "changeSmiley" or "chatMessage" which do not affect the gameflow directly have been added later and may not contained within this file, as they were added in direct correspondence (between Sophia, Pauline).

MainMenu Communication

Messages from Client to Server
- **1.** Attempt Connecting to Server
- **2.** Find game
- **3.** Create Game (List of Players in group, game settings)
- **4.** Send Friend Invite
- **5.** Receive Friend Invite (push + ingame Note)
  - **5.1** Send Accept Friend Invite
  - **5.2** Send Decline Friend Invite
- **6.** Invite friend to group(push + ingame Note)
  - **6.1** Send Accept Invite (push + ingame Note)
  - **6.2** Send Decline Invite (push + ingame Note)

Messages from Server to Client
- **1.** Send Connection Successful
- **2.** Send information about the lobby the client was added to client
- **3.** Send information about the lobby the client was added to client
- **4.** Send friend request to destination client
- **5.** Send information about Friend´s decision to requesting client
  - **5.1** Tell Client that friend approved
  - **5.2** Tell Client that friend declined
- **6.** Send information about Friend´s decision to requesting client
  - **6.1** Tell Client that friend approved
  - **6.2** Tell Client that friend declined

| Client | | Server |

1. Distribute cards to players

Response: {"me": <player object> , "otherPlayers": [<player object1>, <player object2>], "availableCards": [<card object1>, ...] }

Alice Bob Tony

Player Object:

{"cards": [<card object1>, <card object2>, ...], "score": <score>}

score: an integer

2. Alice asks for a new card

Request: {"action": "draw or pick"}

Alice

3. Alice draws a card, e.g. alice.drawCard()

Response: {"action": "draw", "cards": [<card object1, card object2>, ...]},
e.g. alice.getCards() to obtain all cards on Alice hand

3. Send message to Bob & Tony that Alice draws a card
(e.g. Toast message)

{"action": "draw", "playerName": "Alice", "card": <card object>}

Bob Tony

Card object:

{"value": <value>, "type": <type>, "choice": <choice>}

value: 0-13
type: SPADE, DIAMOND, CLUB, HEART
choice: PEEK, SPY, SWAP

4. Alice tries to play (discard?) cards (can be one, two, even three)

Request: {"action": "discard", "cards": [<card object1> , ...]}

Alice

5. Alice discards cards, e.g. alice.discardCards()

Response: {"action": "discard", "cards": [<card object1, card object2>, ...]}

5. Inform Bob & Tony that Alice discarded some cards

Response: {"action": "discard", "cards": [<card object1>, ...], "playerName": "Alice"}

Bob Tony

6. Alice tries to peek a card

{"action": "peek", "card": <card object> }

Alice

7. Inform Bob & Tony that Alice peeked card

{"action": "peek", "card": <card object>, "playerName": "Alice"}

Bob Tony

8. Alice tries to spy Bob's card

{"action": "spy", "card": <card object>, otherPlayer: "Bob" }

Alice

9. Inform Bob & Tony that Alice spied Bob's card

{"action": "spy", "card": <card object>, "playerName": "Alice", spiedPlayer: "Bob" }

Bob Tony

10. Alice tries to swap with Bob's card

{"action": "swap", "myCard": <card object>, "otherCard": <card object>, "otherPlayer": "Bob" }

Alice

11. Alice swaps with Bob's card, e.g. alice.swapWithOtherPlayer()

{"action": "swap", "cards": [<card object1>, ...]}

11. Inform Bob & Tony that Alice swapped Bob's card

{"action": "swap", "card": <card object>, otherCard: <card object>, "playerName": "Alice", swappedPlayer: "Bob" }

Bob Tony