

NET-WizHearts

Erzeugt von Doxygen 1.8.5

Mon Nov 11 2013 23:26:28

Inhaltsverzeichnis

1	Hierarchie-Verzeichnis	1
1.1	Klassenhierarchie	1
2	Klassen-Verzeichnis	4
2.1	Auflistung der Klassen	4
3	Klassen-Dokumentation	8
3.1	ClientMain Klassenreferenz	8
3.1.1	Dokumentation der Elementfunktionen	8
3.2	ClientModel Klassenreferenz	8
3.2.1	Ausführliche Beschreibung	9
3.2.2	Dokumentation der Elementfunktionen	9
3.3	ClientState Enum-Referenz	15
3.4	Card Klassenreferenz	15
3.4.1	Ausführliche Beschreibung	15
3.4.2	Beschreibung der Konstruktoren und Destruktoren	15
3.4.3	Dokumentation der Elementfunktionen	16
3.5	ChooseCards Klassenreferenz	16
3.5.1	Dokumentation der Elementfunktionen	16
3.6	Chooseltem Klassenreferenz	16
3.6.1	Dokumentation der Elementfunktionen	17
3.7	CreateGame Klassenreferenz	17
3.7.1	Ausführliche Beschreibung	18
3.7.2	Dokumentation der Elementfunktionen	18
3.8	DiscardPile Klassenreferenz	18
3.9	DrawDeck Klassenreferenz	18
3.10	Game Klassenreferenz	19
3.10.1	Ausführliche Beschreibung	19
3.10.2	Beschreibung der Konstruktoren und Destruktoren	19
3.10.3	Dokumentation der Elementfunktionen	19
3.11	GameLobby Klassenreferenz	20
3.11.1	Ausführliche Beschreibung	21
3.11.2	Dokumentation der Elementfunktionen	21
3.12	GamePanel Klassenreferenz	23
3.12.1	Ausführliche Beschreibung	23
3.12.2	Dokumentation der Elementfunktionen	23
3.13	InputNumber Klassenreferenz	24
3.13.1	Dokumentation der Elementfunktionen	24
3.14	Language Enum-Referenz	24

3.15	Lobby Klassenreferenz	24
3.15.1	Ausführliche Beschreibung	25
3.15.2	Dokumentation der Elementfunktionen	25
3.16	Login Klassenreferenz	26
3.16.1	Ausführliche Beschreibung	27
3.16.2	Dokumentation der Elementfunktionen	27
3.17	OtherPlayer Klassenreferenz	28
3.18	OwnHand Klassenreferenz	28
3.18.1	Ausführliche Beschreibung	28
3.19	Password Klassenreferenz	28
3.19.1	Dokumentation der Elementfunktionen	29
3.20	ScoreWindow Klassenreferenz	29
3.20.1	Dokumentation der Elementfunktionen	29
3.21	ViewCard Klassenreferenz	30
3.21.1	Ausführliche Beschreibung	30
3.21.2	Beschreibung der Konstruktoren und Destruktoren	30
3.21.3	Dokumentation der Elementfunktionen	30
3.22	Warning Klassenreferenz	31
3.22.1	Ausführliche Beschreibung	31
3.22.2	Dokumentation der Elementfunktionen	31
3.23	ComBeenKicked Klassenreferenz	31
3.23.1	Ausführliche Beschreibung	31
3.23.2	Beschreibung der Konstruktoren und Destruktoren	31
3.23.3	Dokumentation der Elementfunktionen	32
3.24	ComChatMessage Klassenreferenz	32
3.24.1	Ausführliche Beschreibung	32
3.24.2	Beschreibung der Konstruktoren und Destruktoren	32
3.24.3	Dokumentation der Elementfunktionen	32
3.25	ComClientLeave Klassenreferenz	32
3.25.1	Ausführliche Beschreibung	33
3.26	ComClientQuit Klassenreferenz	33
3.26.1	Ausführliche Beschreibung	33
3.27	ComCreateGameRequest Klassenreferenz	33
3.27.1	Ausführliche Beschreibung	33
3.27.2	Beschreibung der Konstruktoren und Destruktoren	33
3.27.3	Dokumentation der Elementfunktionen	34
3.28	ComInitGameLobby Klassenreferenz	34
3.28.1	Ausführliche Beschreibung	35
3.28.2	Beschreibung der Konstruktoren und Destruktoren	35
3.28.3	Dokumentation der Elementfunktionen	35

3.29	ComInitLobby Klassenreferenz	35
3.29.1	Ausführliche Beschreibung	35
3.29.2	Beschreibung der Konstruktoren und Destruktoren	35
3.29.3	Dokumentation der Elementfunktionen	36
3.30	ComJoinRequest Klassenreferenz	36
3.30.1	Ausführliche Beschreibung	36
3.30.2	Beschreibung der Konstruktoren und Destruktoren	36
3.30.3	Dokumentation der Elementfunktionen	37
3.30.4	Dokumentation der Datenelemente	37
3.31	ComKickPlayerRequest Klassenreferenz	37
3.31.1	Ausführliche Beschreibung	37
3.31.2	Beschreibung der Konstruktoren und Destruktoren	37
3.31.3	Dokumentation der Elementfunktionen	38
3.32	ComLobbyUpdateGamelist Klassenreferenz	38
3.32.1	Ausführliche Beschreibung	38
3.32.2	Beschreibung der Konstruktoren und Destruktoren	38
3.32.3	Dokumentation der Elementfunktionen	38
3.33	ComLoginRequest Klassenreferenz	39
3.33.1	Ausführliche Beschreibung	39
3.33.2	Beschreibung der Konstruktoren und Destruktoren	39
3.33.3	Dokumentation der Elementfunktionen	39
3.34	ComRuleset Klassenreferenz	39
3.34.1	Ausführliche Beschreibung	40
3.34.2	Beschreibung der Konstruktoren und Destruktoren	40
3.34.3	Dokumentation der Elementfunktionen	40
3.35	ComServerAcknowledgement Klassenreferenz	40
3.36	ComStartGame Klassenreferenz	40
3.36.1	Ausführliche Beschreibung	40
3.37	ComUpdatePlayerlist Klassenreferenz	41
3.37.1	Ausführliche Beschreibung	41
3.37.2	Beschreibung der Konstruktoren und Destruktoren	41
3.37.3	Dokumentation der Elementfunktionen	41
3.38	ComWarning Klassenreferenz	41
3.38.1	Ausführliche Beschreibung	42
3.38.2	Beschreibung der Konstruktoren und Destruktoren	42
3.38.3	Dokumentation der Elementfunktionen	42
3.39	MsgCard Klassenreferenz	42
3.39.1	Ausführliche Beschreibung	42
3.39.2	Beschreibung der Konstruktoren und Destruktoren	43
3.39.3	Dokumentation der Elementfunktionen	43

3.40	MsgCardRequest Klassenreferenz	43
3.40.1	Ausführliche Beschreibung	43
3.41	MsgGameEnd Klassenreferenz	43
3.41.1	Ausführliche Beschreibung	43
3.42	MsgMultiCards Klassenreferenz	43
3.42.1	Ausführliche Beschreibung	44
3.42.2	Beschreibung der Konstruktoren und Destruktoren	44
3.42.3	Dokumentation der Elementfunktionen	44
3.43	MsgMultiCardsRequest Klassenreferenz	44
3.43.1	Dokumentation der Elementfunktionen	44
3.44	MsgMultipleCardsRequest Klassenreferenz	45
3.45	MsgNumber Klassenreferenz	45
3.45.1	Ausführliche Beschreibung	45
3.45.2	Beschreibung der Konstruktoren und Destruktoren	45
3.45.3	Dokumentation der Elementfunktionen	45
3.46	MsgNumberRequest Klassenreferenz	45
3.46.1	Beschreibung der Konstruktoren und Destruktoren	46
3.47	MsgSelection Klassenreferenz	46
3.47.1	Ausführliche Beschreibung	46
3.47.2	Beschreibung der Konstruktoren und Destruktoren	46
3.47.3	Dokumentation der Elementfunktionen	46
3.48	MsgSelectionRequest Klassenreferenz	46
3.48.1	Ausführliche Beschreibung	47
3.49	MsgUser Klassenreferenz	47
3.49.1	Ausführliche Beschreibung	47
3.49.2	Beschreibung der Konstruktoren und Destruktoren	47
3.49.3	Dokumentation der Elementfunktionen	47
3.50	RulesetMessage Klassenreferenz	47
3.50.1	Ausführliche Beschreibung	48
3.50.2	Dokumentation der Elementfunktionen	48
3.51	Card Schnittstellenreferenz	48
3.51.1	Dokumentation der Elementfunktionen	48
3.52	ClientHearts Klassenreferenz	49
3.52.1	Beschreibung der Konstruktoren und Destruktoren	49
3.52.2	Dokumentation der Elementfunktionen	49
3.53	ClientRuleset Klassenreferenz	50
3.53.1	Ausführliche Beschreibung	50
3.53.2	Beschreibung der Konstruktoren und Destruktoren	50
3.53.3	Dokumentation der Elementfunktionen	51
3.54	ClientWizard Klassenreferenz	53

3.54.1	Beschreibung der Konstruktoren und Destruktoren	53
3.54.2	Dokumentation der Elementfunktionen	54
3.55	Colour Enum-Referenz	54
3.56	GameClientUpdate Klassenreferenz	55
3.56.1	Ausführliche Beschreibung	55
3.56.2	Beschreibung der Konstruktoren und Destruktoren	55
3.56.3	Dokumentation der Elementfunktionen	56
3.57	GamePhase Enum-Referenz	57
3.58	GameState Klassenreferenz	57
3.58.1	Ausführliche Beschreibung	57
3.58.2	Beschreibung der Konstruktoren und Destruktoren	58
3.58.3	Dokumentation der Elementfunktionen	59
3.59	HeartsCard Enum-Referenz	62
3.59.1	Dokumentation der Elementfunktionen	62
3.60	HeartsData Klassenreferenz	62
3.61	OtherData Klassenreferenz	63
3.61.1	Dokumentation der Elementfunktionen	63
3.62	PlayerState Klassenreferenz	64
3.62.1	Ausführliche Beschreibung	64
3.62.2	Beschreibung der Konstruktoren und Destruktoren	64
3.62.3	Dokumentation der Elementfunktionen	65
3.63	RulesetType Enum-Referenz	65
3.64	ServerHearts Klassenreferenz	65
3.64.1	Ausführliche Beschreibung	66
3.64.2	Dokumentation der Elementfunktionen	66
3.65	ServerRuleset Klassenreferenz	66
3.65.1	Ausführliche Beschreibung	67
3.65.2	Beschreibung der Konstruktoren und Destruktoren	68
3.65.3	Dokumentation der Elementfunktionen	69
3.66	ServerWizard Klassenreferenz	73
3.66.1	Ausführliche Beschreibung	74
3.66.2	Dokumentation der Elementfunktionen	74
3.67	WizardCard Enum-Referenz	75
3.67.1	Dokumentation der Elementfunktionen	75
3.68	WizData Klassenreferenz	76
3.68.1	Dokumentation der Elementfunktionen	76
3.69	GameServer Klassenreferenz	76
3.69.1	Ausführliche Beschreibung	77
3.69.2	Beschreibung der Konstruktoren und Destruktoren	77
3.69.3	Dokumentation der Elementfunktionen	77

3.69.4 Dokumentation der Datenelemente	79
3.70 GameServerRepresentation Klassenreferenz	79
3.70.1 Ausführliche Beschreibung	80
3.70.2 Beschreibung der Konstruktoren und Destruktoren	80
3.71 LobbyServer Klassenreferenz	80
3.71.1 Ausführliche Beschreibung	81
3.71.2 Dokumentation der Elementfunktionen	81
3.72 LobbyServer.ClientListenerThread Klassenreferenz	83
3.72.1 Ausführliche Beschreibung	83
3.73 Player Klassenreferenz	83
3.73.1 Ausführliche Beschreibung	84
3.73.2 Beschreibung der Konstruktoren und Destruktoren	84
3.73.3 Dokumentation der Elementfunktionen	84
3.74 Server Klassenreferenz	85
3.74.1 Ausführliche Beschreibung	85
3.74.2 Dokumentation der Elementfunktionen	86
3.75 ServerMain Klassenreferenz	87
3.75.1 Dokumentation der Elementfunktionen	87

Index	88
--------------	-----------

1 Hierarchie-Verzeichnis

1.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

ClientMain	8
ClientModel	8
ClientState	15
Card	15
ChooseCards	16
ChooseItem	16
CreateGame	17
DiscardPile	18
DrawDeck	18
Game	19
GameLobby	20

GamePanel	23
InputNumber	24
Language	24
Lobby	24
Login	26
OtherPlayer	28
OwnHand	28
Password	28
ScoreWindow	29
ViewCard	30
Warning	31
ComBeenKicked	31
ComChatMessage	32
ComClientLeave	32
ComClientQuit	33
ComCreateGameRequest	33
ComInitGameLobby	34
ComInitLobby	35
ComJoinRequest	36
ComKickPlayerRequest	37
ComLobbyUpdateGamelist	38
ComLoginRequest	39
ComRuleset	39
ComServerAcknowledgement	40
ComStartGame	40
ComUpdatePlayerlist	41
ComWarning	41
MsgCardRequest	43
RulesetMessage	47
MsgCard	42
MsgGameEnd	43
MsgMultiCards	43

MsgMultiCardsRequest	44
MsgMultipleCardsRequest	45
MsgNumber	45
MsgNumberRequest	45
MsgSelection	46
MsgSelectionRequest	46
MsgUser	47
Card	48
HeartsCard	62
WizardCard	75
ClientRuleset	50
ClientHearts	49
ClientWizard	53
Colour	54
GameClientUpdate	55
GamePhase	57
GameState	57
OtherData	63
HeartsData	62
WizData	76
PlayerState	64
RulesetType	65
ServerRuleset	66
ServerHearts	65
ServerWizard	73
GameServerRepresentation	79
LobbyServer.ClientListenerThread	83
Player	83
Server	85
GameServer	76
LobbyServer	80
ServerMain	87

2 Klassen-Verzeichnis

2.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

ClientMain	Die ClientMain Klasse startet den Spielclient und initialisiert dessen Komponenten	8
ClientModel	Das ClientModel ist die Schnittstelle zwischen dem MessageListenerThread , dem ClientRuleset und der View	8
ClientState	Dieser Enumerator enthält alle Zustände in denen sich der Client befinden kann	15
Card	Card ist die View-seitige Repräsentation einer Karte	15
ChooseCards	In diesem Fenster muss der Benutzer eine vorbestimmte Menge Karten auswählen	16
ChooseItem	Dieses Fenster ermöglicht es dem Spieler aus einer Liste von Items eines auszuwählen	16
CreateGame	Das Fenster CreateGame dient dem Benutzer zur Erstellung eines neuen Spieles	17
DiscardPile	Stellt einen Ablagestapel dar, dieser kann sowohl für jeden Spieler einzeln oder für alle Spieler gemeinsam in der Mitte des Spielfeldes angezeigt werden	18
DrawDeck	Stellt einen Aufnahmestapel dar	18
Game	Im Game Fenster läuft das Spiel ab. Es enthält den Spielchat und ein GamePanel	19
GameLobby	Die GameLobby modelliert das Wartefenster, in dem beigetretene Spieler auf den Start des Spieles durch den Spielleiter warten	20
GamePanel	Das Panel ist die Komponente des Game-Fensters, welche das eigentliche Spiel darstellt	23
InputNumber	In diesem Fenster, kann der Benutzer eine Zahl eingeben	24
Language	Language stellt Repräsentationen verschiedener Sprachen dar, die von der GUI verwendet werden, um festzustellen welche Anzeigesprache verwendet werden soll	24
Lobby	Diese Klasse erzeugt die Ansicht der ServerLobby auf der Client Seite, in der die Spieler neue Spiele erstellen oder offenen beitreten können	24
Login	Das Login-Fenster repräsentiert den initialen Dialog zwischen Benutzer und Client	26

OtherPlayer	
Zeigt die Informationen über die anderen Spieler an, also den Namen, ein Symbol für die verdeckte Hand und das Label für zusätzliche Angaben	28
OwnHand	
Stellt die Karten dar, die der Spieler auf der Hand hat	28
Password	
Dieses Fenster ermöglicht die Eingabe eines Passwortes um einem Passwortgeschütztem Spiel beizutreten oder per 'Leave' wieder in die Lobby zurückzukehren	28
ScoreWindow	
Dieses Fenster zeigt den momentanen Punktestand nach jeder Runde und den Gesamtpunktestand am Ende des Spieles an	29
ViewCard	
Card ist die View-seitige Repräsentation einer Karte	30
Warning	
Das Warning-Fenster zeigt dem Benutzer Fehlermeldungen bzw	31
ComBeenKicked	
Diese Klasse ist ein spezielles Kommunikations-Objekt	31
ComChatMessage	
Diese Klasse ist ein spezielles Kommunikations-Objekt	32
ComClientLeave	
Diese Klasse ist ein spezielles Kommunikations-Objekt	32
ComClientQuit	
Diese Klasse ist ein spezielles Kommunikations-Objekt	33
ComCreateGameRequest	
Diese Klasse ist ein spezielles Kommunikations-Objekt	33
ComInitGameLobby	
Diese Klasse ist ein spezielles Kommunikations-Objekt	34
ComInitLobby	
Diese Klasse ist ein spezielles Kommunikations-Objekt	35
ComJoinRequest	
Diese Klasse ist ein spezielles Kommunikations-Objekt	36
ComKickPlayerRequest	
Diese Klasse ist ein spezielles Kommunikations-Objekt	37
ComLobbyUpdateGamelist	
Diese Klasse ist ein spezielles Kommunikations-Objekt	38
ComLoginRequest	
Diese Klasse ist ein spezielles Kommunikations-Objekt	39
ComRuleset	
Diese Klasse ist ein spezielles Kommunikations-Objekt	39
ComServerAcknowledgement	
Diese Klasse ist ein spezielles Kommunikations-Objekt	40

ComStartGame	
Diese Klasse ist ein spezielles Kommunikations-Objekt	40
ComUpdatePlayerlist	
Diese Klasse ist ein spezielles Kommunikations-Objekt	41
ComWarning	
Diese Klasse ist ein spezielles Kommunikations-Objekt	41
MsgCard	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	42
MsgCardRequest	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	43
MsgGameEnd	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	43
MsgMultiCards	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	43
MsgMultiCardsRequest	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	44
MsgMultipleCardsRequest	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	45
MsgNumber	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	45
MsgNumberRequest	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	45
MsgSelection	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	46
MsgSelectionRequest	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	46
MsgUser	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	47
RulesetMessage	
Diese Klasse ist eine Verfeinerung der ComRuleset-Klasse	47
Card	
Dieses Interface modelliert eine Spielkarte	48
ClientHearts	
Diese Klasse bildet das Regelwerk für den Clientmodel bei einer Partie Hearts	49
ClientRuleset	
ClientRuleset ist eine abstrakte Klasse und wird zur Regelvorauswertung im Client verwendet	50
ClientWizard	
Diese Klasse bildet das Regelwerk für den Client bei einer Partie Wizard	53
Colour	
Repräsentiert die Farbe einer Karte	54

GameClientUpdate	
Das GameClientUpdate wird vom RuleSet über den GameServer an den Client geschickt und enthält alle Änderungen des GameState , die für den Client relevant sind	55
GamePhase	
Die GamePhase modelliert die verschiedenen Zustände des Spiels im GameState	57
GameState	
Das GameState modelliert einen aktuellen Spielzustand, es wird vom GameServer instanziiert und vom RuleSet bearbeitet	57
HeartsCard	
Modelliert eine Heartskarte,	62
HeartsData	
Die Otherdata eines Spielers zum Spiel Hearts	62
OtherData	
OtherData speichert alle Spielinformationen eines Spielers, außer seiner eigenen Spielhand	63
PlayerState	
Repräsentiert den Spielzustand eines Spielers, und wird unter anderem im GameState gespeichert	64
RulesetType	
Die verschiedenen Regelwerke	65
ServerHearts	
Diese Klasse erstellt das Regelwerk zum Spiel Hearts	65
ServerRuleset	
Das ServerRuleset ist eine abstrakte Klasse und für den Ablauf und die Einhaltung der Regeln eines Spiels zuständig (/L280/)	66
ServerWizard	
Diese Klasse erstellt das Regelwerk zum Spiel Wizard	73
WizardCard	
Modelliert eine Wizardkarte	75
WizData	
Die Otherdata eines Spielers zum Spiel Wizard	76
GameServer	
Diese Klasse ist für die Verwaltung eines Spieles zuständig	76
GameServerRepresentation	
Dies eine Klasse, die Informationen über den Zustand eines Spielers bereithält	79
LobbyServer	
Diese Klasse ist für die Verwaltung der Spiellobby auf dem Server verantwortlich	80
LobbyServer.ClientListenerThread	
Diese innere Klasse ist für das Zustandekommen von Clientverbindungen zuständig	83
Player	
Die Player -Klasse wird zum Versenden von Java Serializable Objects, sowie zum Annehmen solcher verwendet	83
Server	
Ist ein abstrakte Klasse, von der die Klassen LobbyServer und GameServer erben	85

ServerMain

Diese Klasse startet den **Server** und ist für die Konfiguration des Servers verantwortlich

87

3 Klassen-Dokumentation

3.1 ClientMain Klassenreferenz

Öffentliche, statische Methoden

- static void **main** (final String[] args)

Private Attribute

- ClientController **clientController**

3.1.1 Dokumentation der Elementfunktionen

3.1.1.1 static void main (final String[] args) [static]

Parameter

<i>args</i>	
-------------	--

3.2 ClientModel Klassenreferenz

Abgeleitet von Observable.

Öffentliche Methoden

- **ClientModel** ()
- void **leaveWindow** ()
- void **receiveMessage** (ComChatMessage msg)
- void **receiveMessage** (ComInitLobby msg)
- void **receiveMessage** (ComInitGameLobby msg)
- void **receiveMessage** (ComRuleset msg)
- void **receiveMessage** (ComServerAcknowledgement ack)
- void **receiveMessage** (ComBeenKicked msg)
- void **receiveMessage** (ComUpdatePlayerlist update)
- void **receiveMessage** (ComLobbyUpdateGamelist update)
- void **receiveMessage** (ComObject comObject)
- List< String > **getPlayerlist** ()
- List< **GameServerRepresentation** > **getLobbyGamelist** ()
- List< CardID > **getPlayedCards** ()
- List< CardID > **getOwnHand** ()
- List< String > **getOtherPlayerData** ()
- int **getOwnScore** ()
- void **setLanguage** (final **Language** language)
- **Language** **getLanguage** ()
- void **kickPlayer** (final String name)
- void **hostGame** (String gameName, boolean hasPassword, String password, **RulesetType** ruleset)
- void **send** (RulesetMessage msg)
- int **getPlayerCount** ()

- String `getWindowText ()`
- void `setChooseCards (List< Card > cards)`
- void `setInputNumber ()`
- void `sendChatMessage (final String msg)`
- void `joinGame (final String name, final String password)`
- void `startGame ()`
- void `makeMove (CardID card)`
- void `setNetIO (Client.MessageListenerThread netIO)`
- void `createConnection (final String username, final String serverAdress, final int port)`
- String `getWarningText ()`
- `RulesetType[] getRulesets ()`

Private Methoden

- void `sendMessage (ComObject object)`
- void `initGame ()`
- void `informView (ViewNotification note)`

Private Attribute

- String `playerName`
- `ClientRuleset ruleset`
- `Language language`
- `ClientState state`
- `Client.MessageListenerThread netIO`
- `List< String > playerList`
- `Set< GameServerRepresentation > gameList`
- String `chatMessage`

3.2.1 Ausführliche Beschreibung

Das Model prüft Nachrichten, welche es vom MessageListenerThread über die Methode `receiveMessage()` bekommt. RulesetMessages werden an das ClientRuleset weitergeleitet. Weiterhin informiert es seine Observer über Veränderungen und stellt ihnen Methoden zu Verfügung um spielrelevante Daten zu lesen. Weiterhin kann das `ClientModel` ComMessages and den Server schicken, um Kommandos des ClientRulesets oder Eingaben des Controllers weiterzugeben.

3.2.2 Dokumentation der Elementfunktionen

3.2.2.1 void leaveWindow ()

Wird aufgerufen, wenn der User die GameLobby verlässt.

Der Client gelangt zurück in die Lobby.

3.2.2.2 void receiveMessage (ComChatMessage msg)

Sendet eine eingehende Chatnachricht direkt an alle Observer weiter.

Parameter

<code>msg</code>	die ankommende ComChatMessage Nachricht
------------------	---

3.2.2.3 void receiveMessage (ComInitLobby msg)

Diese Methode wird von [receiveMessage\(\)](#) aufgerufen, falls der Server den Spieler erfolgreich in die Lobby hinzugefügt hat.

Empfängt die ComInitGameLobby Nachricht, die eine Liste aller Spieler enthält, die sich in der Lobby befinden. Speichert diese Liste und benachrichtigt die Observer mit der loginSuccessful ViewNotification.

Parameter

<i>msg</i>	die ankommende ComInitLobby Nachricht
------------	---------------------------------------

3.2.2.4 void receiveMessage (ComInitGameLobby msg)

Diese Methode wird von [receiveMessage\(\)](#) aufgerufen, falls der Server den Spieler erfolgreich in die GameLobby hinzugefügt hat.

Empfängt die ComInitGameLobby Nachricht, die eine Liste aller Spieler enthält, die sich in der GameLobby befinden. Speichert diese Liste und benachrichtigt die Observer mit der joinGameSuccessful ViewNotification.

Parameter

<i>msg</i>	die ankommende ComInitGameLobby Nachricht
------------	---

3.2.2.5 void receiveMessage (ComRuleset msg)

Diese Methode wird von [receiveMessage\(\)](#) aufgerufen, falls eine Nachricht für das Regelwerk ankommt.

Die darin enthaltene RulesetMessage wird dem ClientRuleset zur Verarbeitung übergeben.

Parameter

<i>die</i>	ankommende ComRuleset Nachricht
------------	---------------------------------

3.2.2.6 void receiveMessage (ComServerAcknowledgement ack)

Diese Methode wird von [receiveMessage\(\)](#) aufgerufen, falls ein Server Acknowledgement auftritt.

Dabei ist es von Bedeutung, in welchem Zustand sich der Client befindet.

Parameter

<i>ack</i>	Eine Bestätigung durch den Server.
------------	------------------------------------

3.2.2.7 void receiveMessage (ComBeenKicked msg)

Diese Methode wird von [receiveMessage\(\)](#) aufgerufen, falls der Spieler aus der Spiellobby durch einen Spielleiter entfernt wurde.

Der Client gelangt zurück in die Lobby, die Observer werden mit windowChangeForced benachrichtigt.

Parameter

<i>msg</i>	die ankommende ComBeenKicked Nachricht
------------	--

3.2.2.8 void receiveMessage (ComUpdatePlayerlist update)

Diese Methode wird von [receiveMessage\(\)](#) aufgerufen, falls auf dem Server ein neuer Spieler die Lobby/GameLobby betreten hat oder sie von einem Spieler verlassen wurde.

Empfängt die ComUpdatePlayerlist Nachricht, die die Information enthält, ob und welcher Spieler hinzugefügt oder entfernt werden muss. Die Spielerliste wird dementsprechend bearbeitet und die Observer mit playerListUpdate informiert.

Parameter

<i>update</i>	die ankommende ComLobbyUpdatePlayerlist Nachricht
---------------	---

3.2.2.9 void receiveMessage (ComLobbyUpdateGamelist *update*)

Diese Methode wird von [receiveMessage\(\)](#) aufgerufen, falls auf dem Server ein neues Spiel erstellt wurde oder ein Spiel geschlossen/beendet wurde.

Empfängt die ComLobbyUpdateGamelist Nachricht, die die Information enthält, ob und welches Spiel hinzugefügt oder entfernt werden muss. Die Spielliste wird dementsprechend bearbeitet und die Observer mit gameListUpdate informiert.

Parameter

<i>update</i>	die ankommende ComLobbyUpdateGamelist Nachricht
---------------	---

3.2.2.10 void receiveMessage (ComObject *comObject*)

Diese Methode wird von dem ClientListenerThread aufgerufen und bestimmt welche Nachricht sich hinter dem ComObject genau verbirgt um weitere Verarbeitungsschritte einzuleiten.

Parameter

<i>comObject</i>	Die empfangene Nachricht.
------------------	---------------------------

3.2.2.11 List<String> getPlayerlist ()

Liefert eine Liste der Namen der Spieler in der Lobby oder GameLobby.

Rückgabe

Liste von Spielernamen

3.2.2.12 List<GameServerRepresentation> getLobbyGamelist ()

Liefert eine Liste der Spiele, die aktuell auf dem Server offen sind oder gerade gespielt werden.

Rückgabe

Liste aller Spiele der Lobby.

3.2.2.13 List<CardID> getPlayedCards ()

Gibt eine Liste aller bereits ausgespielten Karten zurück.

Rückgabe

enum CardID. Die Ids der Karten

3.2.2.14 List<CardID> getOwnHand ()

Gibt eine Liste der Handkarten des Spielers zurück.

Parameter

<i>Liste</i>	aller Handkarten des Spielers
--------------	-------------------------------

3.2.2.15 List<String> getOtherPlayerData ()

Liefert zusätzliche Daten der anderen Spieler zurück.

Rückgabe

Liste der Stringrepräsentationen der OtherData der Spieler

3.2.2.16 int getOwnScore ()

Gibt den Punktestand des Spielers zurück.

Rückgabe

der eigene Punktestand.

3.2.2.17 void setLanguage (final Language language)

Setzt die Sprache der GUI.

Parameter

<i>language</i>	Enumerator der die Spielsprache anzeigt.
-----------------	--

Benutzt ClientModel.language.

3.2.2.18 Language getLanguage ()

Liefert die Sprache der GUI.

Rückgabe

language Enumerator der die Spielsprache anzeigt.

Benutzt ClientModel.language.

3.2.2.19 void kickPlayer (final String name)

Entfernt einen Spieler aus der GameLobby.

Parameter

<i>Name</i>	des Spielers, der entfernt werden soll
-------------	--

3.2.2.20 void hostGame (String gameName, boolean hasPassword, String password, RulesetType ruleset)

Erstellt ein neues Spiel.

Sendet dazu eine ComCreateGameRequest Nachricht an den Server.

Parameter

<i>gameName</i>	String Name des Spieles.
<i>hasPassword</i>	true, wenn das Spiel ein Passwort hat

<i>password</i>	String Passwort zum sichern des Spieles.
<i>ruleset</i>	das zu verwendende Regelwerk

3.2.2.21 void sendMessage (ComObject *object*) [private]

Sendete erstellte ComObjects an den Server.

Parameter

<i>object</i>	ComObject, das verschickt wird
---------------	--------------------------------

Wird benutzt von ClientModel.send() und ClientModel.sendChatMessage().

3.2.2.22 void send (RulesetMessage *msg*)

Sendet eine RulesetMessage an den Server.

Erstellt dazu eine ComRuleset, die die RulesetMessage enthält.

Parameter

<i>msg</i>	die RulesetMessage, die an den Server geschickt werden soll
------------	---

Benutzt ClientModel.sendMessage().

3.2.2.23 int getPlayerCount ()

Die die Anzahl der Spieler eines Spieles zurück.

Rückgabe

int Die Spielerzahl eines Spieles.

3.2.2.24 String getWindowText ()

Gibt.

Rückgabe

String

3.2.2.25 void sendChatMessage (final String *msg*)

Nimmt vom ClientController eine Chatnachricht entgegen und sendet diese an den Server.

Parameter

<i>msg</i>	die Chatnachricht, die an den Server geschickt werden soll
------------	--

Benutzt ClientModel.sendMessage().

3.2.2.26 void joinGame (final String *name*, final String *password*)

Versucht einem Spiel beizutreten.

Sendet dazu eine ComJoinRequest Nachricht an den Server. Wird diese bestätigt, gelangt der Client in die Game-Lobby. Wird die Nachricht nicht bestätigt, wird eine Fehlermeldung ausgegeben und die Observer mit openWarning informiert.

Parameter

<i>name</i>	String Der Name des Spiels.
<i>password</i>	String Passwort eines Spieles.

3.2.2.27 void startGame ()

Versucht das erstellte Spiel zu starten.

Sendet dazu eine ComStartGame an den Server. Wenn der Client der Spielleiter des Spiels ist, gelangt er ins Spiel. Wenn der Client nicht der Spielleiter des Spiels ist, wird eine Fehlermeldung ausgegeben.

3.2.2.28 void initGame () [private]

Diese Methode wird innerhalb des ClientModels aufgerufen wenn ein Spiel vom Spielleiter gestartet wurde.

Der Client gelangt ins Spiel Die Observer werden über die gameStarted ViewNotification benachrichtigt.

3.2.2.29 void makeMove (CardID card)

Versucht eine Karte auszuspielen.

Lässt dazu vom ClientRuleset überprüfen ob, die ausgewählte Karte gespielt werden darf. Wenn ja, wird sie im ClientRuleset weiterbehandelt. Wenn nein, wird eine Fehlermeldung ausgegeben und dazu die Observer mit open-Warning informiert.

Parameter

<i>die</i>	ID der gespielten Karte
------------	-------------------------

3.2.2.30 void informView (ViewNotification note) [private]

Hilfsmethode die alle verbundenen Observer der GUI kontaktiert.

Parameter

<i>note</i>	Enum der die Art des Aufrufes bestimmt.
-------------	---

3.2.2.31 void createConnection (final String username, final String serverAdress, final int port)

Erstellt den MessageListenerThread und führt den Benutzerlogin durch.

Parameter

<i>username</i>	String der eindeutige Benutzername der für den Login verwendet wird.
<i>serverAdress</i>	String die Adresse des spielervers.
<i>port</i>	Integer der Port des Spielervers.

3.2.2.32 String getWarningText ()

Gibt den Text aus der bei einer Spielwarnung angezeigt wird.

Rückgabe

String Text der Warnung.

3.2.2.33 RulesetType [] getRulesets ()

Liefert ein Array mit allen implementierten Regelwerken.

Parameter

<i>RulesetType[]</i>	Array von unterstützten Regelwerken.
----------------------	--------------------------------------

3.3 ClientState Enum-Referenz

Öffentliche Attribute

- LOGIN
- SERVERLOBBY
- GAMECREATION
- PASSWORDREQUEST
- GAMELOBBY
- GAME
- USERREQUEST
- ENDING

3.4 Card Klassenreferenz

Abgeleitet von JPanel.

Öffentliche Methoden

- [Card](#) (String s, int n)
- int [getID](#) ()
- void **paintComponent** (Graphics g)

Private Attribute

- String **path**
- int **id**
- BufferedImage **face**

Statische, private Attribute

- static final long **serialVersionUID** = 8733682958484899430L

3.4.1 Ausführliche Beschreibung

Sie wird verwendet um einzelne Karten auf das Spielfeld zu zeichnen. Dazu enthält sie die Pfadangabe zu dem Ordner, in dem die Bilder der Karten gespeichert sind, und eine ID, um das genaue Bild zu spezifizieren.

3.4.2 Beschreibung der Konstruktoren und Destruktoren

3.4.2.1 Card (String s, int n)

Erstellt eine neue Karte für die Anzeige und zeichnet dafür das Bild, das durch die Pfadangabe s und seine Kardinalität n im Ordner angegeben ist.

Die Pfadangabe wird durch das Regelwerk bestimmt.

Parameter

<i>s</i>	Pfadangabe zum zu zeichnenden Bild
<i>n</i>	ID der Karte

3.4.3 Dokumentation der Elementfunktionen

3.4.3.1 int getID ()

Gibt die ID der Karte zurÃ¼ck.

Rückgabe

ID der Karte

3.5 ChooseCards Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void [update](#) (Observable o, Object arg)

Private Attribute

- [OwnHand](#) **playerHandPanel**

3.5.1 Dokumentation der Elementfunktionen

3.5.1.1 void update (Observable o, Object arg)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in arg übergebenen Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: openChooseCards, chooseCardsSuccessful

3.6 Chooseltem Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void [update](#) (Observable arg0, Object arg1)

Private Attribute

- Object **itemComboBox**

3.6.1 Dokumentation der Elementfunktionen

3.6.1.1 void update (Observable *arg0*, Object *arg1*)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in *arg1* übergebenen Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: openChooseltem, chooseltemSuccessful

3.7 CreateGame Klassenreferenz

Abgeleitet von JFrame.

Öffentliche Methoden

- [CreateGame](#) ()
- void [addPanelMouseListener](#) (MouseListener m)
- void [addRulesetSelectionListener](#) (ItemListener i)
- void [addCreateButtonListener](#) (ActionListener a)
- void [addLeaveButtonListener](#) (ActionListener a)
- void [setLanguage](#) ([Language](#) l)

Öffentliche, statische Methoden

- static void **main** (String[] args)

Private Methoden

- void **updateLanguage** ()

Private Attribute

- [Language](#) lang
- JTextField **nameField**
- BufferedImage **image**
- JTextField **passwordField**
- JPanel **imagePanel**
- JLabel **lblSelect**
- JComboBox< String > **rulesetBox**
- JCheckBox **chckbxPassword**
- JButton **btnLeave**
- JButton **btnCreate**
- JLabel **lblGameName**

Statische, private Attribute

- static final long **serialVersionUID** = -2893031560688870723L

3.7.1 Ausführliche Beschreibung

Es bietet alle Komponenten, um ein Regelwerk zu wählen, einen Spielnamen festzulegen und das Spiel durch ein Passwort zu schützen. In der Spielerstellung wird ein Titelbild des ausgewählten Spiels und eine kurze Beschreibung angezeigt. Über 'Leave' kehrt der Spieler in die [Lobby](#) zurück und mit 'Create' wird das Spiel erstellt.

3.7.2 Dokumentation der Elementfunktionen

3.7.2.1 void addPanelMouseListener ([MouseListener](#) *m*)

Fügt einen [MouseListener](#) zum [ImagePanel](#) des [CreateGame](#) Fensters hinzu, der zur Anzeige des [MouseOver](#)--Texts verwendet wird.

Parameter

<i>m</i>	ein MouseListener
----------	-----------------------------------

3.7.2.2 void addRulesetSelectionListener ([ItemListener](#) *i*)

Fügt einen [Listener](#) für die Regelwerk-Auswahl des [CreateGame](#) Fensters hinzu.

Parameter

<i>i</i>	ein ItemListener
----------	----------------------------------

3.7.2.3 void addCreateButtonListener ([ActionListener](#) *a*)

Fügt einen [ActionListener](#) für den 'Create' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	------------------------------------

3.7.2.4 void addLeaveButtonListener ([ActionListener](#) *a*)

Fügt einen [ActionListener](#) für den 'Leave' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	------------------------------------

3.7.2.5 void setLanguage ([Language](#) *l*)

Ändert die Sprache des Fensters.

Parameter

<i>l</i>	Sprache in Form des Language -Enums
----------	---

3.8 DiscardPile Klassenreferenz

Private Attribute

- Set< [Card](#) > card

3.9 DrawDeck Klassenreferenz

3.10 Game Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- [Game](#) () throws IOException
- void [makeTrickGameBoard](#) (int playercount)
- void [update](#) (Observable o, Object arg)
- void [update](#) (Observable o, String arg)

Öffentliche, statische Methoden

- static void **main** (String[] args) throws IOException

Private Methoden

- void **makeTrickGameBoardThreePlayers** ()
- void **makeTrickGameBoardFourPlayers** ()
- void **makeTrickGameBoardFivePlayers** ()
- void **makeTrickGameBoardSixPlayers** ()

Private Attribute

- JPanel **contentPane**
- JTextField **textField**

Statische, private Attribute

- static final long **serialVersionUID** = -2655520138213745249L

3.10.1 Ausführliche Beschreibung

Außerdem können über ein Dropdown-Menü Änderungen an Hintergrundbild und Kartenhintergründen vorgenommen werden. Schließen beendet das Spiel und der Spieler wird in die [Lobby](#) zurückgeleitet.

3.10.2 Beschreibung der Konstruktoren und Destruktoren

3.10.2.1 [Game](#) () throws IOException

Erstellt das [Game](#) Fenster.

Ausnahmebehandlung

<i>IOException</i>	
--------------------	--

3.10.3 Dokumentation der Elementfunktionen

3.10.3.1 void [makeTrickGameBoard](#) (int *playercount*)

Arrangiert die Elemente der Spielfeld-Oberfläche für ein Kartenspiel, bei dem Stiche gemacht werden.

Hierfür hat jeder Spieler einen eigenen Ablagestapel vor sich. Es können 3, 4, 5, oder 6 Spieler gewählt werden.

Parameter

<i>playercount</i>	Anzahl der Spieler, wobei 3 <= playercount <=6 einzuhalten ist
--------------------	--

3.10.3.2 void update (Observable o, Object arg)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in arg übergebenen ViewNotification-Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: playedCardsUpdate, otherDataUpdate, moveAcknowledged, gameStarted

3.10.3.3 void update (Observable o, String arg)

Wird durch notify() im [ClientModel](#) aufgerufen, wenn eine Chatnachricht übergeben wird.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet eine Chatnachricht in String-Form

3.11 GameLobby Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- [GameLobby](#) ()
- void [addStartButtonListener](#) (ActionListener a)
- void [addRemoveButtonListener](#) (ActionListener a)
- void [addLeaveButtonListener](#) (ActionListener a)
- void [addChatMessageListener](#) (KeyListener k)
- void [setLanguage](#) ([Language](#) l)
- void [update](#) (Observable o, Object arg)
- void [update](#) (Observable o, String arg)

Öffentliche, statische Methoden

- static void **main** (String[] args)

Private Methoden

- void **updateLanguage** ()

Private Attribute

- JPanel **contentPane**
- JTextField **messageField**
- [Language](#) **lang**
- JButton **btnRemovePlayer**
- JButton **btnLeave**

- JTextArea **chatlog**
- JButton **btnStartGame**

Statische, private Attribute

- static final long **serialVersionUID** = -1899311213351027436L

3.11.1 Ausführliche Beschreibung

Der Spielleiter kann Spieler mit dem Remove Player Button entfernen. Über Leave kehren die Spieler in die Lobby zurück. Der spielinterne Chat ist ab hier verfügbar.

3.11.2 Dokumentation der Elementfunktionen

3.11.2.1 void addStartButtonListener (ActionListener a)

Fügt einen ActionListener für den 'Start Game' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	--------------------

3.11.2.2 void addRemoveButtonListener (ActionListener a)

Fügt einen ActionListener für den 'Remove Player' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	--------------------

3.11.2.3 void addLeaveButtonListener (ActionListener a)

Fügt einen ActionListener für den 'Leave' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	--------------------

3.11.2.4 void addChatMessageListener (KeyListener k)

Fügt einen KeyListener für das Nachricht-Senden-Feld der Lobby hinzu.

Parameter

<i>k</i>	
----------	--

3.11.2.5 void setLanguage (Language l)

Ändert die Sprache des Fensters.

Parameter

<i>l</i>	Sprache in Form des Language-Enums
----------	------------------------------------

3.11.2.6 void update (Observable o, Object arg)

Wird durch notify() im ClientModel aufgerufen.

Je nach dem in `arg` übergebenen `ViewNotification`-Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: joinGameSuccessful, playerListUpdate, windowChangeForced, gameStarted

3.11.2.7 void update (Observable *o*, String *arg*)

Wird aufgerufen, wenn eine String-Nachricht im notify() Ã¼bergeben wird.

Dieser wird als Chatnachricht interpretiert und dem Chatlog angefügt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet einen String, der eine Chatnachricht darstellt

3.12 GamePanel Klassenreferenz

Abgeleitet von JPanel.

Öffentliche Methoden

- [GamePanel](#) ()
- void [setupTrickGame](#) (int players)
- void **paintComponent** (Graphics g)

Private Attribute

- [OwnHand](#) **ownHand**
- Object **ownScoreLabel**
- Set< [OtherPlayer](#) > **otherPlayer**
- [DrawDeck](#) **drawDeck**
- Set< [DiscardPile](#) > **discardPiles**
- BufferedImage **background**

Statische, private Attribute

- static final long **serialVersionUID** = -1041218552426155968L

3.12.1 Ausführliche Beschreibung

Es besteht aus verschiedenen Panelobjekten, welche je nach Regelwerk auf das Spielfeld gezeichnet werden. Dazu gehören die eigenen Karten, eventuell ausgewählte Karten, ein Textfeld z.B. zur Anzeige der Anzahl der restlichen Karten der Mitspieler und den Ablagestapel. Nach jeder Runde wird der Punktestand aktualisiert.

3.12.2 Dokumentation der Elementfunktionen

3.12.2.1 void setupTrickGame (int *players*)

Erzeugt die Komponenten die bei einem Kartenspiel, das um Stiche gespielt wird, für die gewünschte Spielerzahl benötigt werden und ordnet sie an.

Bei diesem Spieltyp erhält jeder Spieler einen eigenen Ablagestapel vor sich.

Parameter

<i>players</i>	Anzahl der Spieler
----------------	--------------------

3.13 InputNumber Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void [update](#) (Observable o, Object arg)

Private Attribute

- Object **numberTextField**

3.13.1 Dokumentation der Elementfunktionen

3.13.1.1 void update (Observable o, Object arg)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in arg übergebenen Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: openInputNumber, inputNumberSuccessful

3.14 Language Enum-Referenz

Öffentliche Attribute

- **German**
- **English**
- **Bavarian**

3.15 Lobby Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- [Lobby](#) ()
- void [addJoinButtonListener](#) (ActionListener a)
- void [addHostButtonListener](#) (ActionListener a)
- void [addLeaveButtonListener](#) (ActionListener a)
- void [addChatMessageListener](#) (KeyListener k)
- void [setLanguage](#) ([Language](#) l)
- void [update](#) (Observable o, Object arg)
- void [update](#) (Observable o, String arg)

Öffentliche, statische Methoden

- static void **main** (String[] args)

Private Methoden

- void **updateLanguage** ()

Private Attribute

- JPanel **contentPane**
- JTextField **messageField**
- JList **playerList**
- JList **gameList**
- JScrollPane **scrollPane**
- JButton **btnHostGame**
- JButton **btnJoinGame**
- JButton **btnLeave**
- JTextArea **chatlog**
- [Language](#) **lang**

Statische, private Attribute

- static final long **serialVersionUID** = 1L

3.15.1 Ausführliche Beschreibung

In der [Lobby](#) werden die Benutzernamen der sich in der [Lobby](#) befindenden Spieler, sowie offene Spiele angezeigt. In der [Lobby](#) können Chatnachrichten gesendet und empfangen werden. Über 'Leave' verlässt der Spieler das Spiel. Über 'Host [Game](#)' wird der Spieler zum CreateGame-Fenster weiter geleitet und mit 'Join [Game](#)' kann einem bereits erstellten Spiel beigetreten werden.

3.15.2 Dokumentation der Elementfunktionen

3.15.2.1 void addJoinButtonListener (ActionListener a)

Fügt einen ActionListener für den 'Join' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	--------------------

3.15.2.2 void addHostButtonListener (ActionListener a)

Fügt einen ActionListener für den 'Host' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	--------------------

3.15.2.3 void addLeaveButtonListener (ActionListener a)

Fügt einen ActionListener für den 'Leave' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	--------------------

3.15.2.4 void addChatMessageListener (KeyListener *k*)

Fügt einen KeyListener für das Nachricht-Senden-Feld der [Lobby](#) hinzu.

Parameter

<i>k</i>	
----------	--

3.15.2.5 void setLanguage (Language *l*)

Ändert die Sprache des Fensters.

Parameter

<i>l</i>	Sprache in Form des Language-Enums
----------	------------------------------------

3.15.2.6 void update (Observable *o*, Object *arg*)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in *arg* übergebenen ViewNotification-Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: joinGameSuccessful, windowChangeForced, playerListUpdate, gameListUpdate, chatMessage

3.15.2.7 void update (Observable *o*, String *arg*)

Wird aufgerufen, wenn eine String-Nachricht im notify() übergeben wird.

Dieser wird als Chatnachricht interpretiert und dem Chatlog angefügt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet einen String, der eine Chatnachricht darstellt

3.16 Login Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- [Login](#) ()
- void [addConnectButtonListener](#) (ActionListener *a*)
- void [addLanguageSelectionListener](#) (ItemListener *i*)
- void [setLanguage](#) (Language *l*)
- void [update](#) (Observable *o*, Object *arg*)

Öffentliche, statische Methoden

- static void **main** (String[] args) throws IOException

Private Methoden

- void **updateLanguage** ()

Private Attribute

- JPanel **contentPane**
- JTextField **nameField**
- JTextField **serverField**
- JComboBox< [Language](#) > **languageComboBox**
- JButton **btnConnect**
- [Language](#) **lang**
- JLabel **lblNickname**
- JLabel **lblHostIp**
- JLabel **lblLanguage**

Statische, private Attribute

- static final long **serialVersionUID** = -2516577977746181978L

3.16.1 Ausführliche Beschreibung

In diesem Fenster kann der Benutzer seinen Namen und die Adresse des Servers eingeben. Außerdem ist über den [Login](#) die Auswahl der Sprache möglich. Über den Login-Button wird die Verbindung zum Server hergestellt.

3.16.2 Dokumentation der Elementfunktionen

3.16.2.1 void addConnectButtonListener (ActionListener a)

Fügt einen Listener für den 'Connect' Button des [Login](#) Fensters hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	--------------------

3.16.2.2 void addLanguageSelectionListener (ItemListener i)

Fügt einen Listener für die Sprachauswahl des [Login](#) Fensters hinzu.

Parameter

<i>i</i>	ein ItemListener
----------	------------------

3.16.2.3 void setLanguage (Language l)

Ändert die Sprache des Fensters.

Parameter

<i>l</i>	Sprache in Form des Language-Enums
----------	------------------------------------

3.16.2.4 void update (Observable o, Object arg)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in arg übergebenen ViewNotification-Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: loginSuccessful

3.17 OtherPlayer Klassenreferenz

Private Attribute

- Object **name**
- Object **info**

3.18 OwnHand Klassenreferenz

Private Attribute

- Object **cards**
- Set< [Card](#) > **card**

3.18.1 Ausführliche Beschreibung

Der Spieler kann eine Karte durch Anklicken auswählen und durch einen zweiten Klick ausspielen.

3.19 Password Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- [Password](#) ()
- void [addJoinButtonListener](#) (ActionListener a)
- void [setLanguage](#) ([Language](#) l)
- void [update](#) (Observable o, Object arg)

Öffentliche, statische Methoden

- static void **main** (String[] args)

Private Methoden

- void **updateLanguage** ()

Private Attribute

- JPanel **contentPane**
- JTextField **textField**
- JButton **btnJoin**
- JLabel **lblEnterPasswordPlease**
- JButton **btnLeave**
- [Language](#) **lang**

Statische, private Attribute

- static final long **serialVersionUID** = 7994797823893327272L

3.19.1 Dokumentation der Elementfunktionen

3.19.1.1 void addJoinButtonListener (ActionListener a)

Fügt einen ActionListener für den 'Join' Button hinzu.

Parameter

<i>a</i>	ein ActionListener
----------	--------------------

3.19.1.2 void setLanguage (Language l)

Ändert die Sprache des Fensters.

Parameter

<i>l</i>	Sprache in Form des Language-Enums
----------	------------------------------------

3.19.1.3 void update (Observable o, Object arg)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in arg übergebenen ViewNotification-Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: openWarning, passwordAccepted

3.20 ScoreWindow Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void [update](#) (Observable o, Object arg)

3.20.1 Dokumentation der Elementfunktionen

3.20.1.1 void update (Observable o, Object arg)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in arg übergebenen Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
----------	--

<i>arg</i>	erwartet: showScore
------------	---------------------

3.21 ViewCard Klassenreferenz

Abgeleitet von JPanel.

Öffentliche Methoden

- [ViewCard](#) (String s, int n)
- int [getID](#) ()
- void **paintComponent** (Graphics g)

Private Attribute

- String **path**
- int **id**
- BufferedImage **face**

Statische, private Attribute

- static final long **serialVersionUID** = 8733682958484899430L

3.21.1 Ausführliche Beschreibung

Sie wird verwendet um einzelne Karten auf das Spielfeld zu zeichnen. Dazu enthält sie die Pfadangabe zu dem Ordner, in dem die Bilder der Karten gespeichert sind, und eine ID, um das genaue Bild zu spezifizieren.

3.21.2 Beschreibung der Konstruktoren und Destruktoren

3.21.2.1 ViewCard (String s, int n)

Erstellt eine neue Karte für die Anzeige und zeichnet dafür das Bild, das durch die Pfadangabe s und seine Kardinalität n im Ordner angegeben ist.

Die Pfadangabe wird durch das Regelwerk bestimmt.

Parameter

<i>s</i>	Pfadangabe zum zu zeichnenden Bild
<i>n</i>	ID der Karte

3.21.3 Dokumentation der Elementfunktionen

3.21.3.1 int getID ()

Gibt die ID der Karte zurück.

Rückgabe

ID der Karte

3.22 Warning Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void [update](#) (Observable o, Object arg)

Private Attribute

- String **warningText**

3.22.1 Ausführliche Beschreibung

Hinweise an, welche vom [ClientModel](#) übergeben wurden. Es wird nur im Fehlerfall angezeigt.

3.22.2 Dokumentation der Elementfunktionen

3.22.2.1 void update (Observable o, Object arg)

Wird durch notify() im [ClientModel](#) aufgerufen.

Je nach dem in arg übergebenen Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

Parameter

<i>o</i>	erwartet ein Objekt von der Klasse ClientModel
<i>arg</i>	erwartet: openWarning

3.23 ComBeenKicked Klassenreferenz

Öffentliche Methoden

- [ComBeenKicked](#) (String message)
- String [getMessage](#) ()

Private Attribute

- String **message**

3.23.1 Ausführliche Beschreibung

Die Nachricht wird an einen Spieler gesendet, wenn er aus einem Spiel entfernt wurde. Dies geschieht, wenn ein Spieler ein Spiel verlässt oder wenn der Spielleiter das Wartefenster verlässt.

3.23.2 Beschreibung der Konstruktoren und Destruktoren

3.23.2.1 ComBeenKicked (String message)

Dies ist der Kontruktor für eine neue ComBeenKicked-Nachricht.

Parameter

<i>message</i>	ist die Nachricht.
----------------	--------------------

3.23.3 Dokumentation der Elementfunktionen**3.23.3.1 String getMessage ()**

Diese Methode liefert die Nachricht, die an den Spieler gesendet wird, wenn er entfernt wird.

Rückgabe

die Nachricht.

3.24 ComChatMessage Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComChatMessage](#) (String message)
- String [getChatMessage](#) ()

Private Attribute

- String **chatMessage**

3.24.1 Ausführliche Beschreibung

Sie enthält eine Chatnachricht in Form eines Strings.

3.24.2 Beschreibung der Konstruktoren und Destruktoren**3.24.2.1 ComChatMessage (String message)**

Dies ist der Kontruktor für eine neue ComChatMessage-Nachricht.

Parameter

<i>message</i>	ist die Chatnachricht, die versendet wird.
----------------	--

3.24.3 Dokumentation der Elementfunktionen**3.24.3.1 String getChatMessage ()**

Hier kann die versendete Nachricht von anderen Klassen ausgelesen werden.

Rückgabe

die Chatnachricht, die versendet wurde.

3.25 ComClientLeave Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComClientLeave](#) ()

3.25.1 Ausführliche Beschreibung

Sie wird zur Benachrichtigung gesendet, wenn ein Spieler ins nächste Fenster möchte und aus dem alten entfernt werden soll.

3.26 ComClientQuit Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComClientQuit](#) ()

3.26.1 Ausführliche Beschreibung

Die Nachricht wird verschickt, wenn der Spieler ein Fenster schließt.

3.27 ComCreateGameRequest Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComCreateGameRequest](#) (String name, Enum ruleset, boolean hasPassword, String password)
- String [getGameName](#) ()
- Enum [getRuleset](#) ()
- boolean [hasPassword](#) ()
- String [getPassword](#) ()

Private Attribute

- String **gameName**
- Enum **ruleset**
- boolean **hasPassword**
- String **password**

3.27.1 Ausführliche Beschreibung

Diese Nachricht wird versendet, wenn ein neues Spiel erstellt werden soll.

3.27.2 Beschreibung der Konstruktoren und Destruktoren

3.27.2.1 ComCreateGameRequest (String name, Enum ruleset, boolean hasPassword, String password)

Dies ist der Kontruktor für eine neue ComCreateGameRequest-Nachricht.

Parameter

<i>name</i>	ist der Name des Spiels.
<i>ruleset</i>	ist die der Spieltyp, der erstellt werden soll.
<i>hasPassword</i>	sagt, ob ein Passwort gesetzt wurde.
<i>password</i>	ist das Passwort, das gesetzt wurde.

Benutzt ComCreateGameRequest.hasPassword().

3.27.3 Dokumentation der Elementfunktionen**3.27.3.1 String getName ()**

Diese Methode gibt den Namen des Spiels zurück.

Rückgabe

den Spielnamen.

3.27.3.2 Enum getRuleset ()

Diese Methode gibt das Regelwerk zurück, das benutzt werden soll.

Rückgabe

das Regelwerk, welches benutzt wird.

3.27.3.3 boolean hasPassword ()

Diese Methode gibt an, ob ein Passwort für ein Spiel gesetzt wurde.

Rückgabe

ob es ein Passwort gibt.

Wird benutzt von ComCreateGameRequest.CreateGameRequest().

3.27.3.4 String getPassword ()

Gibt das Passwort zurück.

Sollte keines gesetzt sein, wird null zurück gegeben.

Rückgabe

das Passwort.

3.28 ComInitGameLobby Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComInitGameLobby](#) (List playerList)
- Object [getPlayerList](#) ()

Private Attribute

- List **playerList**

3.28.1 Ausführliche Beschreibung

Sie liefert die Liste der Spieler, die sich bereits beim Betreten des Wartefensters darin befinden.

3.28.2 Beschreibung der Konstruktoren und Destruktoren

3.28.2.1 ComInitGameLobby (List *playerList*)

Dies ist der Kontruktor für eine neue ComInitGameLobby-Nachricht.

Parameter

<i>playerList</i>	ist die Liste aller Player, die sich im Wartefenster befinden.
-------------------	--

3.28.3 Dokumentation der Elementfunktionen

3.28.3.1 Object getPlayerList ()

Diese Methode gibt die Liste der Player zurück, die sich momentan im Wartefenster befinden.

Rückgabe

die Liste der Spieler.

3.29 ComInitLobby Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComInitLobby](#) (List *playerList*, Set *gameList*)
- List [getPlayerList](#) ()
- Set< [GameServerRepresentation](#) > [getGameList](#) ()

Private Attribute

- List **playerList**
- Set< [GameServerRepresentation](#) > **gameList**

3.29.1 Ausführliche Beschreibung

Sie synchronisiert den Client mit der Lobby, wenn er sich mit dem Server verbindet oder nach einem Spiel in die Lobby zurückkehrt. Dazu enthält sie sowohl die *playerList*, als auch die *gameList*.

3.29.2 Beschreibung der Konstruktoren und Destruktoren

3.29.2.1 ComInitLobby (List *playerList*, Set *gameList*)

Dies ist der Kontruktor für eine neue ComInitLobby-Nachricht.

Parameter

<i>playerList</i>	ist die Liste der Spieler, die sich in der Lobby befinden.
<i>gameList</i>	ist die Liste der Spiele, die existieren und in der Lobby angezeigt werden.

3.29.3 Dokumentation der Elementfunktionen**3.29.3.1 List getPlayerList ()**

Die Methode liefert die Liste aller Spieler, die in der Lobby sind.

Rückgabe

die Liste der Spieler.

3.29.3.2 Set<GameServerRepresentation> getGameList ()

Diese Methode liefert eine Liste aller Spiele, die erstellt wurden, damit sie in der Lobby angezeigt werden können.

Rückgabe

die Liste der Spiele.

3.30 ComJoinRequest Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComJoinRequest](#) (String [gameMasterName](#), String password)
- String [getGameMasterName](#) ()

Private Attribute

- String [gameMasterName](#)
- String **password**

3.30.1 Ausführliche Beschreibung

Sie ist eine Nachricht, die an den Server gesendet wird, wenn der Spieler einem bestimmten Spiel beitreten will. Dazu enthält es den Namen des Spielleiters als String.

3.30.2 Beschreibung der Konstruktoren und Destruktoren**3.30.2.1 ComJoinRequest (String gameMasterName, String password)**

Dies ist der Kontruktor für eine neue ConJoinRequest-Nachricht.

Ein Spiel kann durch den eindeutigen Namen der Spielleiters identifiziert werden.

Parameter

<i>gameMaster-Name</i>	ist der Name der Spielleiters.
<i>String</i>	Passwort für das Spiel.

Benutzt ComJoinRequest.gameMasterName.

3.30.3 Dokumentation der Elementfunktionen

3.30.3.1 String getGameMasterName ()

Diese Methode gibt den Namen des Spielleiters zurück.

Dieser ist eindeutig, so kann ein bestimmtes Spiel identifiziert werden.

Rückgabe

den Namen des Spielleiters.

Benutzt ComJoinRequest.gameMasterName.

3.30.4 Dokumentation der Datenelemente

3.30.4.1 String gameMasterName [private]

Der Name der Spielleiters muss enthalten sein um ein Spiel zuzuordnen.

Der Spielname ist nicht eindeutig, aber der Spielleiter schon. Somit kann jedes Spiel mit Hilfe des Spielleiters identifiziert werden.

Wird benutzt von ComJoinRequest.ComJoinRequest() und ComJoinRequest.getGameMasterName().

3.31 ComKickPlayerRequest Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComKickPlayerRequest](#) (String [playerName](#))
- String [getPlayerName](#) ()

Private Attribute

- String [playerName](#)

3.31.1 Ausführliche Beschreibung

Sie ist eine Nachricht an den Server, die angibt einen Spieler vom Spiel zu entfernen. Dazu enthält es einen String, der den Namen des Spielers enthält.

3.31.2 Beschreibung der Konstruktoren und Destruktoren

3.31.2.1 ComKickPlayerRequest (String playerName)

Dies ist der Konstruktor für eine neue ComKickPlayerRequest-Nachricht.

Diese enthält den Namen des Spielers, der aus dem Spiel gelöscht werden soll.

Parameter

<i>playerName</i>	ist der Name des Spielers.
-------------------	----------------------------

Benutzt ComKickPlayerRequest.playerName.

3.31.3 Dokumentation der Elementfunktionen**3.31.3.1 String getPlayerName ()**

Diese Methode liefert den Namen des Spielers, der aus dem Spiel entfernt werden soll.

Rückgabe

den Spielernamen.

Benutzt ComKickPlayerRequest.playerName.

3.32 ComLobbyUpdateGamelist Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComLobbyUpdateGamelist](#) (boolean removeFlag, [GameServerRepresentation](#) gameServer)
- boolean [isRemoveFlag](#) ()
- [GameServerRepresentation](#) [getGameServer](#) ()

Private Attribute

- boolean **removeFlag**
- [GameServerRepresentation](#) **gameServer**

3.32.1 Ausführliche Beschreibung

Sie aktualisiert die Gameliste in der Lobby. Dazu enthält sie den GameServer und ein RemoveFlag.

3.32.2 Beschreibung der Konstruktoren und Destruktoren**3.32.2.1 ComLobbyUpdateGamelist (boolean removeFlag, GameServerRepresentation gameServer)**

Dies ist der Kontruktor für eine neue ComLobbyUpdateGamelist-Nachricht.

Parameter

<i>removeFlag</i>	zeigt an, ob das Spiel gelöscht werden soll.
<i>gameServer</i>	ist das Spiel.

3.32.3 Dokumentation der Elementfunktionen**3.32.3.1 boolean isRemoveFlag ()**

Diese Methode liefert, ob ein Spiel gelöscht werden soll oder nicht.

Rückgabe

ob das Spiel gelÃ¶scht wird.

3.32.3.2 GameServerRepresentation getGameServer ()

Diese Methode liefert das Spiel, das geupdated werden soll.

Rückgabe

das Spiel.

3.33 ComLoginRequest Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComLoginRequest](#) (String name)
- String [getPlayerName](#) ()

Private Attribute

- String **playerName**

3.33.1 Ausführliche Beschreibung

Sie ist eine Nachricht, die beim Login an den Server gesendet wird. Dazu enthält sie den Namen des Spielers, der sich einloggen möchte.

3.33.2 Beschreibung der Konstruktoren und Destruktoren**3.33.2.1 ComLoginRequest (String name)**

Dies ist der Kontruktor für eine neue ComLoginRequest-Nachricht.

Parameter

<i>name</i>	ist der Name des Spielers, des sich einloggen möchte.
-------------	---

3.33.3 Dokumentation der Elementfunktionen**3.33.3.1 String getPlayerName ()**

Diese Methode liefert den Namen des Spielers, des sich einloggen möchte.

Dieser muss auf Eindeutigkeit geprüft werden.

Rückgabe

den Spielernamen.

3.34 ComRuleset Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComRuleset](#) ([RulesetMessage](#) rulesetMessage)
- [RulesetMessage](#) getRulesetMessage ()

Private Attribute

- [RulesetMessage](#) rulesetMessage

3.34.1 Ausführliche Beschreibung

Sie ist die grundlegende Nachricht eines Regelwerkaufrufes und enthält eine verfeinerte Nachricht mit weiteren Informationen, die [RulesetMessage](#).

3.34.2 Beschreibung der Konstruktoren und Destruktoren

3.34.2.1 ComRuleset (RulesetMessage rulesetMessage)

Dies ist der Kontruktor für eine neue ComResult-Nachricht.

Parameter

<i>rulesetMessage</i>	ist eine Nachricht, die ans Ruleset gesendet werden soll.
-----------------------	---

3.34.3 Dokumentation der Elementfunktionen

3.34.3.1 RulesetMessage getRulesetMessage ()

Diese Methode gibt die Nachricht zurück, die ans Ruleset gesendet werden soll.

Rückgabe

die Nachricht.

3.35 ComServerAcknowledgement Klassenreferenz

Abgeleitet von ComObject.

Weitere Geerbte Elemente

3.36 ComStartGame Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComStartGame](#) ()

3.36.1 Ausführliche Beschreibung

Sie wird versendet, wenn ein Spiel gestartet werden soll.

3.37 ComUpdatePlayerlist Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComUpdatePlayerlist](#) (String playerName, boolean removeFlag)
- String [getPlayerName](#) ()
- boolean [isRemoveFlag](#) ()

Private Attribute

- String **playerName**
- boolean **removeFlag**

3.37.1 Ausführliche Beschreibung

Sie sendet eine Nachricht zum Update der Playerliste in der Lobby und Spiellobby. Dazu enthält sie den Player und ein removeFlag.

3.37.2 Beschreibung der Konstruktoren und Destruktoren

3.37.2.1 ComUpdatePlayerlist (String playerName, boolean removeFlag)

Dies ist der Kontruktor für eine neue ComUpdatePlayerlist-Nachricht.

Diese beinhaltet den Namen des Spielers und die Angabe ob er gelöscht werden soll.

Parameter

<i>playerName</i>	ist der Name der Spielers.
<i>removeFlag</i>	zeigt, ob der Spieler gelöscht werden soll.

3.37.3 Dokumentation der Elementfunktionen

3.37.3.1 String getPlayerName ()

Diese Methode gibt den Namen des Spielers zurück.

Rückgabe

den Spielernamen.

3.37.3.2 boolean isRemoveFlag ()

Diese Methode gibt zurück, ob der Spieler aus der Liste gelöscht werden soll oder nicht.

Rückgabe

ob der Spieler gelöscht werden soll.

3.38 ComWarning Klassenreferenz

Abgeleitet von ComObject.

Öffentliche Methoden

- [ComWarning](#) (String warning)
- String [getWarning](#) ()

Private Attribute

- String **warning**

3.38.1 Ausführliche Beschreibung

Sie soll dem Spieler eine Mitteilung senden und so über ein Fehlerevent informieren.

3.38.2 Beschreibung der Konstruktoren und Destruktoren

3.38.2.1 ComWarning (String warning)

Dies ist der Konstruktor einer neuen ComWarning-Nachricht.

Er enthält eine Warnung an den Spieler, wenn ein Fehler passiert.

Parameter

<i>warning</i>	ist die Warnung, die der Spieler erhält.
----------------	--

3.38.3 Dokumentation der Elementfunktionen

3.38.3.1 String getWarning ()

Diese Methode gibt die Nachricht zurück, die dem Spieler den Fehler mitteilt.

Rückgabe

die Warnnachricht.

3.39 MsgCard Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgCard](#) (Card card)
- Card [getCard](#) ()

Private Attribute

- Card **card**

3.39.1 Ausführliche Beschreibung

Sie beinhaltet die ausgespielte Karte eines Spielers.

3.39.2 Beschreibung der Konstruktoren und Destruktoren

3.39.2.1 MsgCard (Card card)

Dies ist der Kontruktor für eine neue MsgCard-Nachricht.

Diese enthält die Information, welche Karte von einem Spieler gespielt wurde.

Parameter

<i>card</i>	ist die Karte.
-------------	----------------

3.39.3 Dokumentation der Elementfunktionen

3.39.3.1 Card getCard ()

Diese Methode gibt die ausgespielte Karte des Spielers zurück.

Rückgabe

die Karte.

3.40 MsgCardRequest Klassenreferenz

Öffentliche Methoden

- [MsgCardRequest](#) ()

3.40.1 Ausführliche Beschreibung

Diese Nachricht wird von Server gesendet, um einem Spieler mitzuteilen, dass er das Spielen einer Karte erwartet.

3.41 MsgGameEnd Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgGameEnd](#) ()

3.41.1 Ausführliche Beschreibung

Sie signalisiert dem ClientRuleset, dass das Spiel zu Ende ist.

3.42 MsgMultiCards Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgMultiCards](#) (Set cardList)
- Set< [Card](#) > [getCardList](#) ()

Private Attribute

- Set< [Card](#) > **cardList**

3.42.1 Ausführliche Beschreibung

Sie liefert mehrere Karten zum Tausch für das Regelwerk Hearts.

3.42.2 Beschreibung der Konstruktoren und Destruktoren

3.42.2.1 **MsgMultiCards** (Set *cardList*)

Dies ist der Konstruktor für eine neue MsgMultiCards-Nachricht.

Parameter

<i>cardList</i>	ist die Liste der ausgewählten Karten.
-----------------	--

3.42.3 Dokumentation der Elementfunktionen

3.42.3.1 Set<[Card](#)> **getCardList** ()

Gibt die Liste der gewählten Karten zurück.

Rückgabe

die Liste der Karten.

3.43 **MsgMultiCardsRequest** Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgMultiCardsRequest](#) (int *count*)
- int [getCount](#) ()

Private Attribute

- int *count*

3.43.1 Dokumentation der Elementfunktionen

3.43.1.1 int **getCount** ()

Diese Methode gibt die Anzahl der Karten zurück, die der Server vom Spieler erwartet.

Rückgabe

die Anzahl der Karten.

Benutzt [MsgMultiCardsRequest.count](#).

3.44 MsgMultipleCardsRequest Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgMultipleCardsRequest](#) ()

3.45 MsgNumber Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgNumber](#) (int number)
- int [getNumber](#) ()

Private Attribute

- int **number**

3.45.1 Ausführliche Beschreibung

Sie enthält eine Zahl.

3.45.2 Beschreibung der Konstruktoren und Destruktoren

3.45.2.1 [MsgNumber](#) (int *number*)

Dies ist der Kontruktor für eine neue MsgNumber-Nachricht.

Parameter

<i>number</i>	ist eine Eingabe eines Spielers
---------------	---------------------------------

3.45.3 Dokumentation der Elementfunktionen

3.45.3.1 int [getNumber](#) ()

Diese Methode liefert die Eingabe eines Spielers.

Rückgabe

eine Zahl, die Eingabe des Spielers.

3.46 MsgNumberRequest Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgNumberRequest](#) ()

3.46.1 Beschreibung der Konstruktoren und Destruktoren

3.46.1.1 `MsgNumberRequest ()`

<<<<<<< HEAD Dies ist der Kontruktor für eine neue `MsgNumberRequest`-Nachricht.

===== Dies ist der Kontruktor für eine neue `MsgNumberRequest`-Nachricht.

ViewKlassen

3.47 `MsgSelection` Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgSelection](#) (int selection)
- int [getSelection](#) ()

Private Attribute

- int **selection**

3.47.1 Ausführliche Beschreibung

Diese Nachricht enthält Information über eine Auswahl, die der Spieler getroffen hat. Die Wahlmöglichkeiten werden durch Integer repräsentiert.

3.47.2 Beschreibung der Konstruktoren und Destruktoren

3.47.2.1 `MsgSelection (int selection)`

Dies ist der Kontruktor für eine neue `MsgSelection`-Nachricht.

Parameter

<i>selection</i>	ist die getroffene Auswahl, repräsentiert durch einen Integer.
------------------	--

3.47.3 Dokumentation der Elementfunktionen

3.47.3.1 int `getSelection ()`

Diese Methode gibt die Auswahl des Spielers zurück, die er gemacht hat.

Rückgabe

die Auswahl.

3.48 `MsgSelectionRequest` Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgSelectionRequest](#) ()

3.48.1 Ausführliche Beschreibung

Diese Nachricht sendet der Server an einen Spieler, wenn er eine Auswahl von diesem erwartet.

3.49 MsgUser Klassenreferenz

Abgeleitet von [RulesetMessage](#).

Öffentliche Methoden

- [MsgUser](#) ([GameClientUpdate](#) gameClientUpdate)
- [GameClientUpdate](#) [getGameClientUpdate](#) ()

Private Attribute

- [GameClientUpdate](#) **gameClientUpdate**

3.49.1 Ausführliche Beschreibung

Sie wird dem Client gesendet, um dem ClientRuleset den aktuellen Spielzustand in Form eines GameClientUpdate zu übermitteln.

3.49.2 Beschreibung der Konstruktoren und Destruktoren

3.49.2.1 [MsgUser](#) ([GameClientUpdate](#) *gameClientUpdate*)

Dies ist der Konstruktor einer neuen MsgUser-Nachricht.

Parameter

<i>gameClient-Update</i>	ist der aktuelle Spielstand.
--------------------------	------------------------------

3.49.3 Dokumentation der Elementfunktionen

3.49.3.1 [GameClientUpdate](#) [getGameClientUpdate](#) ()

Diese Methode liefert den den aktuellen Spielzustand, der für ein Update benötigt wird.

Rückgabe

den aktuellen Spielzustand.

3.50 RulesetMessage Klassenreferenz

Abgeleitet von [Serializable](#).

Basisklasse für [MsgCard](#), [MsgGameEnd](#), [MsgMultiCards](#), [MsgMultiCardsRequest](#), [MsgMultipleCardsRequest](#), [MsgNumber](#), [MsgNumberRequest](#), [MsgSelection](#), [MsgSelectionRequest](#) und [MsgUser](#).

Öffentliche Methoden

- void [visit](#) ([ServerRuleset](#) serverRuleset, String name)
- void [visit](#) ([ClientRuleset](#) clientRuleset)

3.50.1 Ausführliche Beschreibung

Sie enthält einen Nachrichtentyp und vererbt an alle Nachrichten für das Regelwerk.

3.50.2 Dokumentation der Elementfunktionen

3.50.2.1 void visit (*ServerRuleset serverRuleset*, *String name*)

Diese Methode ist nötig, damit das *ServerRuleset* entscheiden kann welche Message es enthält und wie diese verarbeitet werden soll.

Parameter

<i>serverRuleset</i>	ist das Ruleset, welches übergeben wird, damit die überladene Methode richtig gewählt wird.
<i>name</i>	ist der Name des Spielers.

3.50.2.2 void visit (*ClientRuleset clientRuleset*)

Diese Methode ist nötig, damit das *ServerRuleset* entscheiden kann welche Message es enthält und wie diese verarbeitet werden soll.

Parameter

<i>clientRuleset</i>	ist das Ruleset, welches übergeben wird, damit die überladene Methode richtig gewählt wird.
----------------------	---

3.51 Card Schnittstellenreferenz

Basisklasse für [HeartsCard](#) und [WizardCard](#).

Öffentliche Methoden

- int [getValue](#) ()
- [Colour](#) [getColour](#) ()

3.51.1 Dokumentation der Elementfunktionen

3.51.1.1 int getValue ()

Gibt den Wert der Karte zurück.

Rückgabe

Der Wert der Karte

Implementiert in [WizardCard](#) und [HeartsCard](#).

3.51.1.2 Colour getColour ()

Gibt die Farbe der Karte zurück.

Rückgabe

Die Farbe der Karte

Implementiert in [WizardCard](#) und [HeartsCard](#).

3.52 ClientHearts Klassenreferenz

Abgeleitet von [ClientRuleset](#).

Öffentliche Methoden

- [ClientHearts](#) ([ClientModel](#) client)
- boolean [isValidMove](#) ([Card](#) card)
- void [send](#) (Set< [Card](#) > cards)
- void [resolveMessage](#) ([MsgMultipleCardsRequest](#) msgMultiCardsRequest)

Statische, private Attribute

- static final int [MIN_PLAYERS](#) = 4
- static final int [MAX_PLAYERS](#) = 4

Weitere Geerbte Elemente

3.52.1 Beschreibung der Konstruktoren und Destruktoren

3.52.1.1 [ClientHearts](#) ([ClientModel](#) client)

Erzeugt ein [ClientHearts](#).

Parameter

<i>client</i>	Das Model auf dem gespielt wird
---------------	---------------------------------

Benutzt [ClientHearts.MAX_PLAYERS](#) und [ClientHearts.MIN_PLAYERS](#).

3.52.2 Dokumentation der Elementfunktionen

3.52.2.1 boolean [isValidMove](#) ([Card](#) card) [virtual]

Überprüft ob ein gemachter Zug zu dem Spiel Hearts gültig ist.

Rückgabe

[isValid](#) true falls Zug gültig, false wenn nicht

Implementiert [ClientRuleset](#).

3.52.2.2 void [send](#) (Set< [Card](#) > cards)

Schickt ein Set an Karten an den Server.

Parameter

<i>cards</i>	Das Set an Karten
--------------	-------------------

3.52.2.3 void [resolveMessage](#) ([MsgMultipleCardsRequest](#) msgMultiCardsRequest)

Verarbeitet die RulesetMessage dass der Server von dem Spieler verlangt mehrere Karten anzugeben.

Parameter

<i>msgMultiCards-Request</i>	Die Nachricht vom Server
------------------------------	--------------------------

3.53 ClientRuleset Klassenreferenz

Basisklasse für [ClientHearts](#) und [ClientWizard](#).

Öffentliche Methoden

- [RulesetType](#) [getRulesetType](#) ()
- int [getMinPlayers](#) ()
- int [getMaxPlayers](#) ()
- [GamePhase](#) [getGamePhase](#) ()
- List< [Card](#) > [getOwnHand](#) ()
- [OtherData](#) [getOwnData](#) ()
- [OtherData](#) [getOtherPlayerData](#) (String player)
- [PlayerState](#) [getCurrentPlayer](#) ()
- [Card](#) [getTrumpCard](#) ()
- void [resolveMessage](#) ([RulesetMessage](#) message)
- void [resolveMessage](#) ([MsgUser](#) clientUpdate)
- void [resolveMessage](#) ([MsgCardRequest](#) msgCardRequest)
- void [send](#) ([Card](#) card)
- abstract boolean [isValidMove](#) ([Card](#) card)

Geschützte Methoden

- [ClientRuleset](#) ([RulesetType](#) ruleset, int minPlayers, int maxPlayers, [ClientModel](#) client)
- void [send](#) ([RulesetMessage](#) message)

Private Attribute

- [ClientModel](#) client
- [GameClientUpdate](#) gameState
- final [RulesetType](#) RULESET
- final int MIN_PLAYERS
- final int MAX_PLAYERS
- [GamePhase](#) gamePhase

3.53.1 Ausführliche Beschreibung

Dazu benutzt es die [isValidMove\(\)](#) Methode. Des Weiteren kann es vom [ClientModel](#) erhaltene [RulesetMessages](#) mit der [resolveMessage\(\)](#) Methode behandeln und neue [RulesetMessages](#) senden.

3.53.2 Beschreibung der Konstruktoren und Destruktoren

3.53.2.1 [ClientRuleset](#) ([RulesetType](#) ruleset, int minPlayers, int maxPlayers, [ClientModel](#) client) [protected]

Erstellt eine [ClientRuleset](#) Klasse.

Parameter

<i>ruleset</i>	Das Ruleset zum Spiel
<i>minPlayers</i>	Die minimale Spieleranzahl
<i>maxPlayers</i>	Die maximale Spieleranzahl
<i>client</i>	Das ClientModel auf dem gespielt wird

Benutzt ClientRuleset.client, ClientRuleset.gamePhase, ClientRuleset.MAX_PLAYERS, ClientRuleset.MIN_PLAYERS, ClientRuleset.RULESET und GamePhase.Start.

3.53.3 Dokumentation der Elementfunktionen

3.53.3.1 RulesetType getRulesetType ()

Gibt den Typ des Regelwerks zurück.

Rückgabe

Der Typ vom Regelwerk

Benutzt ClientRuleset.RULESET.

3.53.3.2 int getMinPlayers ()

Gibt die Mindestanzahl an Spielern zurück für dieses Spiel.

Rückgabe

Die Mindestanzahl an Spielern

Benutzt ClientRuleset.MIN_PLAYERS.

3.53.3.3 int getMaxPlayers ()

Gibt die Maximale Anzahl an Spielern zurück.

Rückgabe

Die maximale Anzahl an Spielern

Benutzt ClientRuleset.MAX_PLAYERS.

3.53.3.4 GamePhase getGamePhase ()

Gibt die momentanen Spielphase zurück.

Rückgabe

gamePhase Die Spielphase

Benutzt ClientRuleset.gamePhase.

3.53.3.5 List<Card> getOwnHand ()

Gibt die eigenen Handkarten zurück.

Rückgabe

Liste von Karten

3.53.3.6 OtherData getOwnData ()

Gibt die **OtherData** des Models zurück.

Rückgabe

Die Otherdata des Models

3.53.3.7 OtherData getOtherPlayerData (String *player*)

Holt die **OtherData** eines anderen Spielers.

Parameter

<i>Der</i>	Spielername
------------	-------------

Rückgabe

otherPlayerData Die **OtherData**

3.53.3.8 PlayerState getCurrentPlayer ()

Gibt den Spieler der momentan am Zug ist zurück.

Rückgabe

Der momentane Spieler

3.53.3.9 Card getTrumpCard ()

Holt die aufgedeckte Trumpfkarte.

Rückgabe

Eine Karte

3.53.3.10 void resolveMessage (RulesetMessage *message*)

Verarbeitet eine RulesetMessage vom Server.

Parameter

<i>clientUpdate</i>	Die Nachricht vom Server
---------------------	--------------------------

3.53.3.11 void resolveMessage (MsgUser *clientUpdate*)

Verarbeitet die RulesetMessage dass der Server ein Spielupdate an den Client schickt.

Parameter

<i>clientUpdate</i>	Die Nachricht vom Server
---------------------	--------------------------

3.53.3.12 void resolveMessage (MsgCardRequest *msgCardRequest*)

Verarbeitet die RulesetMessage dass der Server von dem Spieler verlangt eine Karte zu spielen.

Parameter

<i>msgCard-Request</i>	Die Nachricht vom Server
------------------------	--------------------------

3.53.3.13 void send (Card card)

Verpackt eine Karte in eine Rulesetmessage und schickt sie an den Server.

Parameter

<i>card</i>	Die karte
-------------	-----------

3.53.3.14 void send (RulesetMessage message) [protected]

Schickt eine RulesetMessage übers Model an den Server.

Parameter

<i>message</i>	Die Nachricht
----------------	---------------

3.53.3.15 abstract boolean isValidMove (Card card) [pure virtual]

Prüft ob ein gemachter Zug in einem Spiel gültig war.

Parameter

<i>card</i>	Die Karte
-------------	-----------

Implementiert in [ClientHearts](#) und [ClientWizard](#).

3.54 ClientWizard Klassenreferenz

Abgeleitet von [ClientRuleset](#).

Öffentliche Methoden

- boolean [isValidMove](#) (Card card)
- void [send](#) (int number)
- void [send](#) (Colour colour)
- void [resolveMessage](#) (MsgNumberRequest msgNumber)
- void [resolveMessage](#) (MsgSelectionRequest msgSelection)

Geschützte Methoden

- [ClientWizard](#) (ClientModel client)

Statische, private Attribute

- static final int [MIN_PLAYERS](#) = 3
- static final int [MAX_PLAYERS](#) = 6

3.54.1 Beschreibung der Konstruktoren und Destruktoren

3.54.1.1 ClientWizard (ClientModel client) [protected]

Erzeugt ein [ClientWizard](#).

Parameter

<i>client</i>	Das Model auf dem gespielt wird
---------------	---------------------------------

Benutzt ClientWizard.MAX_PLAYERS und ClientWizard.MIN_PLAYERS.

3.54.2 Dokumentation der Elementfunktionen**3.54.2.1 boolean isValidMove (Card card) [virtual]**

Prüft ob ein gemachter Zug zum Spiel Wizard gültig ist.

Parameter

<i>card</i>	Eine gespielte Karte
-------------	----------------------

Rückgabe

isValid true falls Zug gültig, false wenn nicht

Implementiert [ClientRuleset](#).

3.54.2.2 void send (int number)

Sendet die Anzahl der angesagten Stiche.

Parameter

<i>number</i>	Die Anzahl der angesagten Stiche
---------------	----------------------------------

Wird benutzt von ClientWizard.send().

3.54.2.3 void send (Colour colour)

Sendet eine ausgewählte Trumpffarbe.

Parameter

<i>colour</i>	Die Trumpffarbe
---------------	-----------------

Benutzt ClientWizard.send().

3.54.2.4 void resolveMessage (MsgNumberRequest msgNumber)

Verarbeitet die RulesetMessage dass der Server von dem Spieler verlangt eine Stichanzahl anzugeben.

Parameter

<i>msgNumber</i>	Die Nachricht vom Server
------------------	--------------------------

3.54.2.5 void resolveMessage (MsgSelectionRequest msgSelection)

Verarbeitet die RulesetMessage dass der Server von dem Spieler verlangt eine Farbe auszuwählen.

Parameter

<i>msgSelection</i>	Die Nachricht vom Server
---------------------	--------------------------

3.55 Colour Enum-Referenz**Öffentliche Attribute**

- **NONE**

- HEART
- CLUB
- SPADE
- DIAMOND
- BLUE
- RED
- YELLOW

3.56 GameClientUpdate Klassenreferenz

Geschützte Methoden

- [GameClientUpdate](#) ([PlayerState](#) *playerState*, [Map](#)< [String](#), [Card](#) > *discardPile*, [Map](#)< [String](#), [OtherData](#) > *otherPlayerData*, [PlayerState](#) *currentPlayer*, [Card](#) *trumpCard*)
- [List](#)< [Card](#) > [getOwnHand](#) ()
- [Map](#)< [String](#), [Card](#) > [getPlayedCards](#) ()
- [OtherData](#) [getOwnData](#) ()
- [OtherData](#) [getOtherPlayerData](#) ([String](#) *player*)
- [PlayerState](#) [getCurrentPlayer](#) ()
- [Card](#) [getTrumpCard](#) ()

Private Attribute

- [PlayerState](#) *playerState*
- [Map](#)< [String](#), [Card](#) > *discardPile*
- [Map](#)< [String](#), [OtherData](#) > *otherPlayerData*
- [PlayerState](#) *currentPlayer*
- [Card](#) *trumpCard*

3.56.1 Ausführliche Beschreibung

Das wären seine Spielhand, der Ablagestapel sowie die Otherdata von allen Spielern und die Trumpfkarte.

3.56.2 Beschreibung der Konstruktoren und Destruktoren

3.56.2.1 [GameClientUpdate](#) ([PlayerState](#) *playerState*, [Map](#)< [String](#), [Card](#) > *discardPile*, [Map](#)< [String](#), [OtherData](#) > *otherPlayerData*, [PlayerState](#) *currentPlayer*, [Card](#) *trumpCard*) [protected]

Erstellt ein [GameClientUpdate](#).

Parameter

<i>playerState</i>	Der Spielerzustand des Client
<i>discardPile</i>	Der Ablagestapel
<i>otherPlayerData</i>	Die Daten der anderen Spieler
<i>currentPlayer</i>	Der momentan aktive Spieler
<i>trumpCard</i>	Die Trumpffarbe

Benutzt [GameClientUpdate.currentPlayer](#), [GameClientUpdate.discardPile](#), [GameClientUpdate.otherPlayerData](#), [GameClientUpdate.playerState](#) und [GameClientUpdate.trumpCard](#).

3.56.3 Dokumentation der Elementfunktionen

3.56.3.1 `List<Card> getOwnHand ()` [protected]

Holt die Karten die der Client auf der Hand hat.

Rückgabe

ownHand Die Hand des Clients

3.56.3.2 `Map<String, Card> getPlayedCards ()` [protected]

Holt die gespielten Karten auf dem Ablagestapel.

Rückgabe

discardPile Die gespielten Karten

Benutzt GameClientUpdate.discardPile.

3.56.3.3 `OtherData getOwnData ()` [protected]

Holt die Otherdata des Client als String als Stringrepräsentation.

Rückgabe

ownData Die Otherdata des Clients

3.56.3.4 `OtherData getOtherPlayerData (String player)` [protected]

Holt die [OtherData](#) eines anderen Spielers als Stringrepräsentation.

Parameter

<i>player</i>	Der Name des Spielers
---------------	-----------------------

Rückgabe

otherPlayerData Die [OtherData](#) der anderen Spieler

3.56.3.5 `PlayerState getCurrentPlayer ()` [protected]

Gibt den Spieler der momentan am Zug ist zurück.

Rückgabe

Der momentane Spieler

Benutzt GameClientUpdate.currentPlayer.

3.56.3.6 `Card getTrumpCard ()` [protected]

Holt die aufgedeckte Trumpfkarte.

Rückgabe

trumpCard Die Trumpfkarte

Benutzt GameClientUpdate.trumpCard.

3.57 GamePhase Enum-Referenz

Öffentliche Attribute

- [Start](#)
- [Playing](#)
- [CardRequest](#)
- [MultipleCardRequest](#)
- [TrickRequest](#)
- [SelectionRequest](#)
- [Ending](#)

3.58 GameState Klassenreferenz

Geschützte Methoden

- [GameState](#) ([RulesetType](#) ruleset, List< [Card](#) > deck)
- boolean [addPlayerToGame](#) (String name)
- void [setFirstPlayer](#) ([PlayerState](#) player)
- [PlayerState](#) [getFirstPlayer](#) ()
- boolean [setCurrentPlayer](#) ([PlayerState](#) player)
- [PlayerState](#) [getCurrentPlayer](#) ()
- List< [Card](#) > [getCardsLeftInDeck](#) ()
- Map< String, [Card](#) > [getPlayedCards](#) ()
- [PlayerState](#) [getPlayerState](#) (String name)
- void [setTrumpCard](#) ([Card](#) trumpCard)
- [Card](#) [getTrumpCard](#) ()
- int [getRoundNumber](#) ()
- int [getNumberOfPlayedCards](#) ()
- List< [Card](#) > [getPlayerCards](#) (String name)
- void [shuffleDeck](#) ()
- boolean [dealCards](#) (int number)
- void [nextPlayer](#) ()
- boolean [giveACard](#) (String name, [Card](#) card)
- boolean [playCard](#) ([Card](#) card)

Private Attribute

- List< [PlayerState](#) > [players](#)
- [RulesetType](#) ruleset
- [PlayerState](#) firstPlayer
- [PlayerState](#) currentPlayer
- int [roundNumber](#)
- Map< String, [Card](#) > [discardPile](#)
- List< [Card](#) > [deck](#)
- [Card](#) trumpCard

3.58.1 Ausführliche Beschreibung

Es enthält die einzelnen PlayerStates, sowie Informationen zum Ablage-, Aufnahmestapel, Rundenanzahl, den momentan aktiven Spieler sowie [GamePhase](#).

3.58.2 Beschreibung der Konstruktoren und Destruktoren

3.58.2.1 **GameState** (**RulesetType** *ruleset*, List< **Card** > *deck*) [protected]

Erstellt eine GameStateklasse und trumpCard wird als WizardCard.Empty instanziiert.

Parameter

<i>ruleset</i>	Der Regelwerktyp des Spiels
<i>deck</i>	Das Kartendeck im Spiel

Benutzt GameState.deck, GameState.discardPile, GameState.players, GameState.ruleset und GameState.trumpCard.

3.58.3 Dokumentation der Elementfunktionen

3.58.3.1 boolean addPlayerToGame (String *name*) [protected]

Fügt den Spieler ins Spiel hinein, falls er nicht schon im Spiel ist.

Parameter

<i>name</i>	Der Name eines Spielers
-------------	-------------------------

Rückgabe

true falls der Spieler noch nicht im Spiel ist, und false sonst

Benutzt GameState.players und GameState.ruleset.

3.58.3.2 void setFirstPlayer (**PlayerState** *player*) [protected]

Setzt einen neuen Spieler als firstPlayer.

Parameter

<i>player</i>	Der neue firstPlayer
---------------	----------------------

Benutzt GameState.firstPlayer.

3.58.3.3 **PlayerState** getFirstPlayer () [protected]

Holt den Spieler der als erster am Zug war.

Rückgabe

firstPlayer Der Spielzustand des Spielers der als erster am Zug war

Benutzt GameState.firstPlayer.

3.58.3.4 boolean setCurrentPlayer (**PlayerState** *player*) [protected]

Setzt einen neuen Spieler als currentPlayer.

Parameter

<i>player</i>	Der neue currentPlayer
---------------	------------------------

Benutzt GameState.currentPlayer.

3.58.3.5 **PlayerState** getCurrentPlayer () [protected]

Holt den Spieler der momentan am Zug ist.

Rückgabe

currentPlayer Der Spielzustand des Spielers der grad am Zug ist

Benutzt GameState.currentPlayer.

3.58.3.6 `List<Card> getCardsLeftInDeck ()` [protected]

Holt die Karten die noch im Aufnahmestapel sind.

Rückgabe

deck Holt die Karten die noch im Aufnahmestapel sind

3.58.3.7 `Map<String,Card> getPlayedCards ()` [protected]

Holt die gespielten Karten im Ablagestapel.

Rückgabe

discardPile Die gespielten Karten

Benutzt GameState.discardPile.

3.58.3.8 `PlayerState getPlayerState (String name)` [protected]

Holt einen bestimmten Spieler.

Parameter

<i>name</i>	Der Name des Spielers
-------------	-----------------------

Rückgabe

player Der Spielzustand des Spielers

Benutzt GameState.players.

3.58.3.9 `void setTrumpCard (Card trumpCard)` [protected]

Setzt die Trumpfkarte.

Parameter

<i>trumpCard</i>	Die Trumpfkarte
------------------	-----------------

Benutzt GameState.trumpCard.

3.58.3.10 `Card getTrumpCard ()` [protected]

Holt die momentane Trumpfkarte im Spiel.

Rückgabe

trumpCard Die momentane Trumpfkarte

Benutzt GameState.trumpCard.

3.58.3.11 `int getRoundNumber ()` [protected]

Holt die Anzahl an Runden.

Rückgabe

Die Anzahl der Runden

Benutzt GameState.roundNumber.

3.58.3.12 `int getNumberOfPlayedCards () [protected]`

Holt die Anzahl der gespielten Karten.

Rückgabe

Die Anzahl der gespielten Karten

3.58.3.13 `List<Card> getPlayerCards (String name) [protected]`

Holt die Karten eines Spielers.

Parameter

<i>name</i>	Der Name vom Spieler
-------------	----------------------

Rückgabe

Die Karten eines Spielers zurück, wenn der Spieler nicht gefunden wird, wird eine leere Liste zurückgegeben

Benutzt GameState.players.

3.58.3.14 `boolean dealCards (int number) [protected]`

Verteilt eine bestimmte Anzahl an Karten an die Spieler.

Parameter

<i>number</i>	Die Anzahl an Karten
---------------	----------------------

Rückgabe

True falls ein Spieler keine Karten hat, false sonst

Benutzt GameState.deck und GameState.players.

3.58.3.15 `boolean giveACard (String name, Card card) [protected]`

Gibt eine bestimmte Karte einem Spieler.

Parameter

<i>name</i>	Der Name des Spielers
<i>card</i>	Die Karte

Rückgabe

true falls die Karte im Stapel ist, false wenn nicht

3.58.3.16 `boolean playCard (Card card) [protected]`

Entfernt eine Karte aus der Hand des currentPlayer und legt sie auf dem Ablagestapel.

Parameter

<i>card</i>	Die gespielte Karte
-------------	---------------------

Rückgabe

isInHand Gibt true zurück wenn die gespielte Karte auf der Hand vom Spieler liegt und false sonst

3.59 HeartsCard Enum-Referenz

Abgeleitet von [Card](#).

Öffentliche Methoden

- int [getValue](#) ()
- [Colour](#) [getColour](#) ()

Öffentliche Attribute

- **Empty** =(0,Colour.NONE)
- **Herz2** =(0,Colour.HEART)
- **Caro3** =(3,Colour.DIAMOND)

Private Methoden

- **HeartsCard** (int value, [Colour](#) colour)

Private Attribute

- final int **value**
- final [Colour](#) **colour**

3.59.1 Dokumentation der Elementfunktionen

3.59.1.1 int [getValue](#) ()

Gibt den Wert der Karte zurück.

Rückgabe

Der Wert der Karte

Implementiert [Card](#).

3.59.1.2 [Colour](#) [getColour](#) ()

Gibt die Farbe der Karte zurück.

Rückgabe

Die Farbe der Karte

Implementiert [Card](#).

3.60 HeartsData Klassenreferenz

Abgeleitet von [OtherData](#).

Öffentliche Methoden

- String [toString](#) ()

Geschützte Methoden

- [HeartsData](#) ()

3.61 OtherData Klassenreferenz

Basisklasse für [HeartsData](#) und [WizData](#).

Öffentliche Methoden

- abstract String [toString](#) ()

Geschützte Methoden

- [OtherData](#) ()
- void [madeTrick](#) (Set< [Card](#) > tricks)
- List< [Card](#) > [removeTricks](#) ()
- int [getNumberOfTricks](#) ()
- void [setPoints](#) (int [points](#))
- int [getPoints](#) ()

Private Attribute

- Set< [Card](#) > [madeTricks](#)
- int [points](#)

3.61.1 Dokumentation der Elementfunktionen

3.61.1.1 void [madeTrick](#) (Set< [Card](#) > *tricks*) [protected]

Gibt dem Spieler die Stichkarten die er gemacht.

Parameter

<i>tricks</i>	Die Stiche
---------------	------------

3.61.1.2 List<[Card](#)> [removeTricks](#) () [protected]

Entfernt die gemachten Stichkarten eines Spielers und fügt sie wieder in den Kartenstapel.

Rückgabe

Die Kartenstiche

Benutzt [OtherData.madeTricks](#).

3.61.1.3 int [getNumberOfTricks](#) () [protected]

Gibt die Anzahl der gemachten Stiche des Spielers zurück.

Rückgabe

Die Anzahl der gemachten Stiche

3.61.1.4 void [setPoints](#) (int *points*) [protected]

Setzt den Punktestand eines Spielers.

Parameter

<i>points</i>	Der neue Punktestand
---------------	----------------------

Benutzt OtherData.points.

3.61.1.5 `int getPoints ()` [protected]

Gibt den Punktestand eines Spielers zurück.

Rückgabe

Der Punktestand

Benutzt OtherData.points.

3.62 PlayerState Klassenreferenz**Öffentliche Methoden**

- [PlayerState](#) (String *name*, [RulesetType](#) ruleset)

Geschützte Methoden

- String [getName](#) ()
- List< [Card](#) > [getHand](#) ()
- [OtherData](#) [getOtherData](#) ()
- void [addCard](#) ([Card](#) card)
- boolean [removeCard](#) ([Card](#) card)

Private Attribute

- String [name](#)
- List< [Card](#) > [ownHand](#)
- [OtherData](#) [otherData](#)

3.62.1 Ausführliche Beschreibung

Sie enthält den Namen des Spielers, seine Handkarten und [OtherData](#).

3.62.2 Beschreibung der Konstruktoren und Destruktoren**3.62.2.1 [PlayerState](#) (String *name*, [RulesetType](#) ruleset)**

Erstellt einen [PlayerState](#).

Parameter

<i>name</i>	Der Name des Spielers
<i>ruleset</i>	Der Typ des Spiels

Benutzt [PlayerState.name](#) und [PlayerState.otherData](#).

3.62.3 Dokumentation der Elementfunktionen

3.62.3.1 String getName () [protected]

Holt den Namen eines Spielers.

Rückgabe

name Der Name des Spielers

3.62.3.2 List<Card> getHand () [protected]

Holt die Kartenhand des Spielers.

Rückgabe

ownHand Die Kartenhand des Spielers

3.62.3.3 OtherData getOtherData () [protected]

Gibt die [OtherData](#) des Spielers zurück.

Rückgabe

otherData Die [OtherData](#) eines Spielers

Benutzt PlayerState.otherData.

3.62.3.4 void addCard (Card card) [protected]

Gibt dem Spieler eine Karte.

Parameter

<i>card</i>	Die Karte die dem Spieler gegeben wird
-------------	--

3.62.3.5 boolean removeCard (Card card) [protected]

Entfernt eine Karte aus der Hand des Spielers.

Parameter

<i>card</i>	
-------------	--

Rückgabe

ownHand.remove(card) Gibt true zurück wenn die Karte in der Hand ist und false sonst

3.63 RulesetType Enum-Referenz

Öffentliche Attribute

- **Wizard**
- **Hearts**

3.64 ServerHearts Klassenreferenz

Abgeleitet von [ServerRuleset](#).

Öffentliche Methoden

- [ServerHearts](#) ([GameServer](#) s)

Geschützte Methoden

- boolean [isValidMove](#) ([Card](#) card)
- void [resolveMessage](#) ([MsgMultiCards](#) msgMultiCard, String name)
- void [calculateRoundOutcome](#) ()
- void [calculateTricks](#) ()
- String [getWinner](#) ()

Statische, private Attribute

- static final int **MIN_PLAYERS** = 4
- static final int **MAX_PLAYERS** = 4

3.64.1 Ausführliche Beschreibung

Sie enthält zudem weitere Methoden, welche für das Spiel Hearts spezifisch benötigt werden, wie die Regelung zum Tausch von Karten und die Berechnung der Stichpunkten.

3.64.2 Dokumentation der Elementfunktionen

3.64.2.1 boolean isValidMove ([Card](#) card) [protected], [virtual]

Prüft ob ein gemachter Zug vom currentPlayer in einem Spiel gültig war, wenn nicht wird an den Spieler erneut eine [MsgCardRequest](#) gesendet.

Parameter

<i>card</i>	Die Karte die gespielt wurde
-------------	------------------------------

Rückgabe

true falls Zug gültig und false wenn nicht

Implementiert [ServerRuleset](#).

3.64.2.2 void resolveMessage ([MsgMultiCards](#) msgMultiCard, String name) [protected]

Verarbeitet die RulesetMessage dass mehrere Karten von einem Spieler übergeben wurden.

Parameter

<i>msgMultiCard</i>	Die Nachricht vom Client
<i>name</i>	Der Name des Spielers

3.65 ServerRuleset Klassenreferenz

Basisklasse für [ServerHearts](#) und [ServerWizard](#).

Öffentliche Methoden

- [RulesetType](#) [getRulesetType](#) ()
- int [getMinPlayers](#) ()
- int [getMaxPlayers](#) ()
- [GamePhase](#) [getGamePhase](#) ()
- void [runGame](#) ()

Geschützte Methoden

- [ServerRuleset](#) ([RulesetType](#) ruleset, int min, int max, [GameServer](#) server)
- void [setFirstPlayer](#) ([PlayerState](#) player)
- [PlayerState](#) [getFirstPlayer](#) ()
- boolean [nextPlayer](#) ()
- boolean [setCurrentPlayer](#) ([PlayerState](#) player)
- [OtherData](#) [getOtherData](#) ([PlayerState](#) player)
- [PlayerState](#) [getCurrentPlayer](#) ()
- void [addPlayerToGame](#) (String name)
- [PlayerState](#) [getPlayerState](#) (String name)
- List< [Card](#) > [getPlayerCards](#) (String name)
- void [send](#) ([RulesetMessage](#) message, String name)
- void [broadcast](#) ([RulesetMessage](#) message)
- void [resolveMessage](#) ([MsgCard](#) msgCard, String name)
- boolean [dealCards](#) (int number)
- boolean [giveACard](#) (String name, [Card](#) card)
- boolean [playCard](#) ([Card](#) card)
- void [setTrumpCard](#) ([Card](#) card)
- abstract boolean [isValidMove](#) ([Card](#) card)
- abstract void [calculateTricks](#) ()
- abstract void [calculateRoundOutcome](#) ()
- abstract String [getWinner](#) ()

Private Methoden

- List< [Card](#) > [createDeck](#) ()

Private Attribute

- [GameServer](#) server
- [GameState](#) gameState
- [GamePhase](#) gamePhase
- final [RulesetType](#) RULESET
- final int MIN_PLAYERS
- final int MAX_PLAYERS

3.65.1 Ausführliche Beschreibung

Das [ServerRuleset](#) wird im [GameServer](#) instanziiert und verwaltet die Zustände des [GameStates](#) im Server. Mit der Methode [isValidMove\(\)](#) wird eine Eingabe eines Clients auf Regelkonformität überprüft und dann im [GameServer](#) das [GameState](#) verändert. Über [resolveMessage\(\)](#) kann eine [GameServer](#)instanz eine [RulesetMessage](#) vom Player an das Ruleset weiterleiten.

3.65.2 Beschreibung der Konstruktoren und Destruktoren

3.65.2.1 **ServerRuleset** (**RulesetType** *ruleset*, int *min*, int *max*, **GameServer** *server*) [protected]

Erstellt ein [ServerRuleset](#).

Parameter

<i>ruleset</i>	Der Rulesettyp vom Server
<i>min</i>	Die minimale Anzahl an Spielern
<i>max</i>	Die maximale Anzahl an Spielern
<i>server</i>	Der Server auf dem gespielt wird

Benutzt `ServerRuleset.gamePhase`, `ServerRuleset.MAX_PLAYERS`, `ServerRuleset.MIN_PLAYERS`, `ServerRuleset.RULESET`, `ServerRuleset.server` und `GamePhase.Start`.

3.65.3 Dokumentation der Elementfunktionen

3.65.3.1 **RulesetType** `getRulesetType ()`

Gibt den Typ des Regelwerks zurück.

Rückgabe

Der Typ vom Regelwerk

Benutzt `ServerRuleset.RULESET`.

3.65.3.2 **int** `getMinPlayers ()`

Gibt die Mindestanzahl an Spielern zurück für dieses Spiel.

Rückgabe

Die Mindestanzahl an Spielern

Benutzt `ServerRuleset.MIN_PLAYERS`.

3.65.3.3 **int** `getMaxPlayers ()`

Gibt die Maximale anzahl an Spielern zurück.

Rückgabe

Die maximale Anzahl an Spielern

Benutzt `ServerRuleset.MAX_PLAYERS`.

3.65.3.4 **GamePhase** `getGamePhase ()`

Gibt den momentanen Spielzustand zurück.

Rückgabe

Gibt die momentan Spielphase zurück

Benutzt `ServerRuleset.gamePhase`.

3.65.3.5 **List<Card>** `createDeck ()` [private]

Erzeugt ein Kartendeck, abhängig von dem [RulesetType](#).

Rückgabe

Gibt ein Kartendeck zurück

Benutzt `ServerRuleset.RULESET`.

3.65.3.6 void setFirstPlayer (**PlayerState** *player*) [protected]

Setzt den Spieler der als Erster am Zug ist, im Gamestate.

Parameter

<i>Der</i>	Spielerzustand des Spielers
------------	-----------------------------

3.65.3.7 PlayerState getFirstPlayer () [protected]

Holt den Spieler der als erster am Zug war.

Rückgabe

firstPlayer Der Spielzustand des Spielers der als erster am Zug war

3.65.3.8 boolean nextPlayer () [protected]

Setzt den nächsten Spieler in der List als currentPlayer.

Rückgabe

true falls es ein anderer Spieler ist und false wenn es derselbe ist.

3.65.3.9 boolean setCurrentPlayer (PlayerState player) [protected]

Setzt den Spieler der am Nächsten am Zug ist, im Gamestate.

Parameter

<i>player</i>	Der Playerstate eines Spielers
---------------	--------------------------------

Rückgabe

false wenn der selbe Spieler nochmal als currentPlayer gesetzt wird

3.65.3.10 OtherData getOtherData (PlayerState player) [protected]

Die [OtherData](#) eines Spielers.

Parameter

<i>player</i>	Der Spielerzustand
---------------	--------------------

Rückgabe

Gibt [OtherData](#) zurück

3.65.3.11 PlayerState getCurrentPlayer () [protected]

Holt den Spieler der gerade am Zug ist.

Rückgabe

currentPlayer Der Spielzustand des Spielers der grad am Zug ist

3.65.3.12 void addPlayerToGame (String name) [protected]

Fügt einen Spieler ins Spiel ein.

Parameter

<i>name</i>	Der name vom Spieler
-------------	----------------------

3.65.3.13 `PlayerState getPlayerState (String name)` [protected]

Holt den Spielerzustand eines Spielers.

Parameter

<i>name</i>	Der Name des Spielers
-------------	-----------------------

Rückgabe

playerState Spielzustand eines Spielers

3.65.3.14 `List<Card> getPlayerCards (String name)` [protected]

Holt die Spielkarten eines Spielers.

Parameter

<i>name</i>	Der Name eines Spielers
-------------	-------------------------

Rückgabe

Die Spielkarten des Spielers

3.65.3.15 `void send (RulesetMessage message, String name)` [protected]

Schickt eine Nachricht an einen Spieler.

Parameter

<i>message</i>	Die Nachricht vom Typ RulesetMessage
<i>name</i>	Der Name vom Spieler

3.65.3.16 `void broadcast (RulesetMessage message)` [protected]

Schickt eine Nachricht an alle Spieler.

Parameter

<i>message</i>	Die Nachricht
----------------	---------------

3.65.3.17 `void resolveMessage (MsgCard msgCard, String name)` [protected]

Verarbeitet die RulesetMessage dass eine Karte vom Spieler gespielt.

Parameter

<i>msgCard</i>	Die Nachricht vom Client welche Karte gespielt wurde
<i>name</i>	Der Name des Spielers

3.65.3.18 `boolean dealCards (int number)` [protected]

Verteilt eine bestimmte Anzahl an Karten an die Spieler.

Parameter

<i>number</i>	Die Anzahl an Karten
---------------	----------------------

Rückgabe

Gibt true zurück wenn ein Spieler keine Karten hat, false sonst

3.65.3.19 `boolean giveACard (String name, Card card)` [protected]

Gibt einem Spieler eine bestimmte Karte.

Parameter

<i>name</i>	Der Name eines Spielers
<i>card</i>	Eine Karte

Rückgabe

Gibt true zurück wenn die Karte im Deck ist, false sonst

3.65.3.20 `boolean playCard (Card card)` [protected]

Der momentane Spieler spielt eine Karte.

Parameter

<i>card</i>	Die gespielte Karte
-------------	---------------------

Rückgabe

true falls der Spieler die Karte hat

3.65.3.21 `void setTrumpCard (Card card)` [protected]

Setzt eine Karte als Trumpf.

Parameter

<i>card</i>	Eine karte
-------------	------------

3.65.3.22 `abstract boolean isValidMove (Card card)` [protected],[pure virtual]

Prüft ob ein gemachter Zug vom currentPlayer in einem Spiel gültig war, wenn nicht wird an den Spieler erneut eine MsgCardRequest gesendet.

Parameter

<i>card</i>	Die Karte die gespielt wurde
-------------	------------------------------

Rückgabe

true falls Zug gültig und false wenn nicht

Implementiert in [ServerWizard](#) und [ServerHearts](#).

3.66 ServerWizard Klassenreferenz

Abgeleitet von [ServerRuleset](#).

Öffentliche Methoden

- [ServerWizard](#) ([GameServer](#) s)

Geschützte Methoden

- boolean [isValidMove](#) ([Card](#) card)
- void [calculateRoundOutcome](#) ()
- void [resolveMessage](#) ([MsgNumber](#) msgNumber, String name)
- void [resolveMessage](#) ([MsgSelection](#) msgSelection, String name)
- void [calculateTricks](#) ()
- String [getWinner](#) ()

Statische, private Attribute

- static final int **MIN_PLAYERS** = 3
- static final int **MAX_PLAYERS** = 6

3.66.1 Ausführliche Beschreibung

Sie enthält zudem weitere Methoden, welche für das Spiel Wizard spezifisch benötigt werden, wie das Ansage von Stichen, der Bestimmung von Trumpffarben und die Bestimmung der Rundenanzahl.

3.66.2 Dokumentation der Elementfunktionen

3.66.2.1 boolean [isValidMove](#) ([Card](#) *card*) [protected], [virtual]

Prüft ob ein gemachter Zug vom `currentPlayer` in einem Spiel gültig war, wenn nicht wird an den Spieler erneut eine `MsgCardRequest` gesendet.

Parameter

<i>card</i>	Die Karte die gespielt wurde
-------------	------------------------------

Rückgabe

true falls Zug gültig und false wenn nicht

Implementiert [ServerRuleset](#).

3.66.2.2 void [resolveMessage](#) ([MsgNumber](#) *msgNumber*, String *name*) [protected]

Verarbeitet die `RulesetMessage` dass ein Spieler eine Stichangabe gemacht hat.

Parameter

<i>msgNumber</i>	Die Nachricht vom Client
<i>name</i>	Der Name des Spielers

3.66.2.3 void [resolveMessage](#) ([MsgSelection](#) *msgSelection*, String *name*) [protected]

Verarbeitet die `RulesetMessage` dass ein Spieler eine Farbe ausgewählt hat.

Parameter

<i>msgSelection</i>	Die Nachricht vom Client
<i>name</i>	Der Name des Spielers

3.67 WizardCard Enum-Referenz

Abgeleitet von [Card](#).

Öffentliche Methoden

- [int](#) [getValue](#) ()
- [Colour](#) [getColour](#) ()

Öffentliche Attribute

- **Empty** =(0,Colour.NONE)
- **NarrBlau** =(0,Colour.BLUE)
- **ZaubererRot** =(14,Colour.RED)
- **EinsGruen** =(1,Colour.GREEN)
- **ZweiGruen** =(2,Colour.GREEN)
- **DreiGruen** =(3,Colour.GREEN)
- **ZweiRot** =(2,Colour.RED)
- **DreiRot** =(3,Colour.RED)
- **VierRot** =(4,Colour.RED)

Private Methoden

- **WizardCard** (int value, [Colour](#) colour)

Private Attribute

- final int **value**
- final [Colour](#) **colour**

3.67.1 Dokumentation der Elementfunktionen

3.67.1.1 [int](#) [getValue](#) ()

Gibt den Wert der Karte zurück.

Rückgabe

Der Wert der Karte

Implementiert [Card](#).

3.67.1.2 [Colour](#) [getColour](#) ()

Gibt die Farbe der Karte zurück.

Rückgabe

Die Farbe der Karte

Implementiert [Card](#).

3.68 WizData Klassenreferenz

Abgeleitet von [OtherData](#).

Öffentliche Methoden

- String [toString](#) ()

Geschützte Methoden

- [WizData](#) ()
- int [getAnnouncedTricks](#) ()
- void [setAnnounceTricks](#) (int annouceTricks)

Private Attribute

- int [announcedTricks](#)

3.68.1 Dokumentation der Elementfunktionen

3.68.1.1 int [getAnnouncedTricks](#) () [[protected](#)]

Holt die angesagten Stiche des Spielers.

Rückgabe

[announcedTricks](#) Die angesagten Stiche

3.68.1.2 void [setAnnounceTricks](#) (int *annouceTricks*) [[protected](#)]

Beim Spielstart werden die vorausgesagten Stiche des Spieler gespeichert.

Parameter

<i>annouceTricks</i>	Die vorausgesagten Stiche des Spielers
----------------------	--

3.69 GameServer Klassenreferenz

Abgeleitet von [Server](#).

Öffentliche Methoden

- [GameServer](#) ([LobbyServer](#) server, [Player](#) gameMaster, String GameName, [RulesetType](#) ruleset, String [password](#), boolean [hasPassword](#))
- [GameServerRepresentation](#) [getRepresentation](#) ()
- synchronized void [addPlayer](#) ([Player](#) player)
- synchronized void [removePlayer](#) ([Player](#) player)
- void [receiveMessage](#) ([Player](#) player, [ComKickPlayerRequest](#) kickPlayer)
- void [receiveMessage](#) ([Player](#) player, [ComChatMessage](#) chat)
- void [receiveMessage](#) ([Player](#) player, [ComClientLeave](#) leave)
- void [receiveMessage](#) ([Player](#) player, [ComClientQuit](#) quit)
- void [receiveMessage](#) ([Player](#) player, [ComStartGame](#) start)
- void [receiveMessage](#) ([Player](#) player, [ComRuleset](#) [ruleset](#))
- [ComInitGameLobby](#) [initLobby](#) ()
- void [handleIOException](#) ([Player](#) player)

Private Attribute

- [LobbyServer](#) lobbyServer
- String [gameMasterName](#)
- String [name](#)
- String [password](#)
- int [maxPlayers](#)
- int [currentPlayers](#)
- boolean [hasPassword](#)
- [RulesetType](#) rulesetType
- [ServerRuleset](#) ruleset

Weitere Geerbte Elemente

3.69.1 Ausführliche Beschreibung

Sie verwaltet die Kommunikation zwischen den Clients während eines Spieles. Die GameServer-Klasse erbt Methoden zur Kommunikation vom [Server](#). Der [GameServer](#) tauscht Nachrichten zwischen Ruleset und [Player](#) aus, um so den Spielablauf zu koordinieren.

3.69.2 Beschreibung der Konstruktoren und Destruktoren

3.69.2.1 [GameServer](#) ([LobbyServer](#) server, [Player](#) gameMaster, String GameName, [RulesetType](#) ruleset, String password, boolean hasPassword)

Konstruktor des GameServers.

Setzt die Attribute lobbyServer, name, password, hasPasword und rulesetType auf die übergebenen Werte. Setzt den gameMasterName auf den Namen des gameMaster und fügt den gameMaster dem Set an Spielern hinzu. Bestimmt mithilfe des Enums RulesetType das Ruleset und erstellt es. Setzt currentPlayers auf eins und maxPlayers je nach Ruleset.

Parameter

<i>server</i>	ist der LobbyServer der den GameServer erstellt hat.
<i>gameMaster</i>	ist der Name des Spielleiters
<i>GameName</i>	ist der Name des Spiels
<i>ruleset</i>	gibt an, welches Ruleset verwendet wird
<i>password</i>	speichert das Passwort des Spiels
<i>hasPassword</i>	gibt an, ob das Spiel ein Passwort hat

3.69.3 Dokumentation der Elementfunktionen

3.69.3.1 synchronized void addPlayer ([Player](#) player)

Diese Methode fügt einen [Player](#) dem Set an Playern hinzu, welche der [Server](#) verwaltet.

Es wird vorausgesetzt, dass der [Player](#) gültig und noch nicht im Set vorhanden ist. Zusätzlich wird die Zahl der currentPlayers um eins Erhöht.

Parameter

<i>player</i>	ist der Player , der hinzugefügt wird
---------------	---

Benutzt [GameServer.currentPlayers](#).

3.69.3.2 synchronized void removePlayer (Player *player*)

Diese Methode entfernt einen **Player** aus dem Set an Playern, welche der **Server** verwaltet.

Es wird vorausgesetzt, dass der **Player** gültig und im Set vorhanden ist. Zusätzlich wird die Zahl der currentPlayers um eins Verringert.

Parameter

<i>player</i>	ist der Player , der entfernt wird
---------------	---

Benutzt GameServer.currentPlayers.

3.69.3.3 void receiveMessage (Player *player*, ComKickPlayerRequest *kickPlayer*)

Diese Methode ist dafür zuständig zu ermitteln, was passiert wenn ein Spieler aus der GameLobby geworfen wird.

Der **Player** wird durch Aufruf von changeServer an die Lobby zurückgegeben. An diesen Spieler wird ein ComInitLobby und ein ComWarning geschickt. Danach wird ein ComUpdatePlayerlist Objekt mit broadcast an alle Clients im Spiel verschickt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>kicked</i>	ist das ComObject, das verarbeitet wird

3.69.3.4 void receiveMessage (Player *player*, ComChatMessage *chat*)

Diese Methode ist dafür zuständig eine Chatnachricht an alle Clients im Spiel zu verschicken.

Dafür wird die ComChatMessage mit broadcast an alle Spieler im playerSet verteilt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>chat</i>	ist das ComObject, das die Chatnachricht enthält

Benutzt Server.broadcast().

3.69.3.5 void receiveMessage (Player *player*, ComClientLeave *leave*)

Diese Methode gibt einen **Player**, der die GameLobby verlassen will, durch Aufruf von changeServer an die ServerLobby zurück und schickt ihm ein ComInitLobby.

Danach wird ein ComUpdatePlayerlist Objekt mit broadcast an alle Clients im Spiel verschickt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>leave</i>	ist das ComObject, welches angibt, dass der Spieler in die Lobby zurückkehrt

3.69.3.6 void receiveMessage (Player *player*, ComClientQuit *quit*)

Diese Methode behandelt den Fall, dass ein Spieler das laufende Spiel verlässt.

Sie gibt einen **Player**, der das Spiel verlassen will, Aufruf von changeServer an die ServerLobby zurück und schickt ihm ein ComInitLobby. Alle Spieler, die sich im Spiel befinden werden durch Aufruf von changeServer an die Lobby zurückgegeben und bekommen ein ComInitLobby und ein ComWarning. Das Spiel wird aus dem gameServerSet des LobbyServers entfernt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>quit</i>	ist das ComObject, welches angibt, dass der Spieler das Spiel verlässt

3.69.3.7 void receiveMessage (Player *player*, ComStartGame *start*)

Diese Methode sagt dem Ruleset, dass ein neues Spiel gestartet werden soll indem er dessen runGame Methode aufruft.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>start</i>	ist das ComObject, dass angibt, dass das Spiel gestartet werden soll

3.69.3.8 void receiveMessage (Player *player*, ComRuleset *ruleset*)

Diese Methode gibt das erhaltene ComRuleset durch einen Aufruf von resolveMessage an das Ruleset weiter.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>ruleset</i>	ist das ComObject, das zeigt, dass das Object vom Ruleset bearbeitet werden muss

3.69.3.9 ComInitGameLobby initLobby ()

Baut ein neues ComInitGameLobby Objekt und gibt es zurück.

Rückgabe

Gibt das ComInitGameLobby Objekt zurück

3.69.3.10 void handleIOException (Player *player*)

Diese Methode legt den Ablauf fest, was passiert, falls die Verbindung zu einem Client verloren gegangen ist.

Der übergebene [Player](#) wird aus dem playerSet im [GameServer](#), sowie dem names Set im [LobbyServer](#) entfernt. Alle Spieler, die sich im Spiel befinden werden durch Aufruf von changeServer an die Lobby zurückgegeben und bekommen ein ComInitLobby und ein ComWarning. Das Spiel wird aus dem gameServerSet des LobbyServers entfernt.

Parameter

<i>player</i>	ist der Tread von dem die IOException kommt
---------------	---

3.69.4 Dokumentation der Datenelemente

3.69.4.1 String password [private]

Das Passwort, das der Spielleiter beim erstellen gesetzt hat.

Ist NULL, falls es kein Passwort gibt.

3.70 GameServerRepresentation Klassenreferenz

Öffentliche Methoden

- [GameServerRepresentation](#) (String gameMaster, String gameName, int max, int current, [RulesetType](#) type, boolean password)
- String [getGameMasterName](#) ()
- void [setGameMasterName](#) (String [gameMasterName](#))

- String [getName](#) ()
- void [setName](#) (String [name](#))
- int [getMaxPlayers](#) ()
- void [setMaxPlayers](#) (int [maxPlayers](#))
- int [getCurrentPlayers](#) ()
- void [setCurrentPlayers](#) (int [currentPlayers](#))
- [RulesetType](#) [getRuleset](#) ()
- void [setRuleset](#) ([RulesetType](#) [ruleset](#))
- boolean [isHasPassword](#) ()
- void [setHasPassword](#) (boolean [hasPassword](#))

Private Attribute

- String [gameMasterName](#)
- String [name](#)
- int [maxPlayers](#)
- int [currentPlayers](#)
- [RulesetType](#) [ruleset](#)
- boolean [hasPassword](#)

3.70.1 Ausführliche Beschreibung

Sie wird dem ComObjekt [ComLobbyUpdateGameList](#) angehängt, um die Spielliste in der [GameLobby](#) aktualisieren zu können

3.70.2 Beschreibung der Konstruktoren und Destruktoren

3.70.2.1 [GameServerRepresentation](#) (String [gameMaster](#), String [gameName](#), int [max](#), int [current](#), [RulesetType](#) [type](#), boolean [password](#))

Der Konstruktor der Klasse [GameServerRepresentation](#) initialisiert die Attribute mit den vom [GameServer](#) übergebenen Werten.

Parameter

<i>gameMaster</i>	der Name des Spielleiters
<i>gameName</i>	der Name des Spiels
<i>max</i>	Maximal mögliche Anzahl teilnehmender Spieler
<i>current</i>	Anzahl momentaner Spieler
<i>type</i>	Welches Ruleset verwendet wird
<i>password</i>	ob das Spiel ein Passwort hat

3.71 LobbyServer Klassenreferenz

Abgeleitet von [Server](#).

Klassen

- class [ClientListenerThread](#)

Öffentliche Methoden

- [LobbyServer](#) ()
- void [addName](#) (String name)
- void [removeName](#) (String name)
- void [addGameServer](#) ([GameServer](#) game)
- void [removeGameServer](#) ([GameServer](#) game)
- void [receiveMessage](#) ([Player](#) player, ComChatMessage chat)
- void [receiveMessage](#) ([Player](#) player, ComClientQuit quit)
- void [receiveMessage](#) ([Player](#) player, ComCreateGameRequest create)
- void [receiveMessage](#) ([Player](#) player, ComJoinRequest join)
- void [receiveMessage](#) ([Player](#) player, ComLoginRequest login)
- ComInitLobby [initLobby](#) ()
- void [handleIOException](#) ([Player](#) player)

Private Attribute

- Set< String > [names](#)
- Set< [Player](#) > [noNames](#)
- Set< [GameServer](#) > [gameServerSet](#)
- [ClientListenerThread](#) [clientListenerThread](#)
- ServerSocket [socket](#)

Weitere Geerbte Elemente

3.71.1 Ausführliche Beschreibung

Sie erstellt neue Spiele und verwaltet laufende Spiele. Auch wird der Chatverkehr über sie an die verbundenen Spieler weitergeleitet. Die LobbyServer-Klasse erbt Methoden zur Kommunikation vom [Server](#).

3.71.2 Dokumentation der Elementfunktionen

3.71.2.1 void addName (String name)

Fügt einen neuen Benutzennamen in das Namensset ein.

Es wird vorausgesetzt, dass der Name noch nicht im Set vorhanden ist.

Parameter

<i>name</i>	ist der Name der eingefügt wird
-------------	---------------------------------

3.71.2.2 void removeName (String name)

Löscht einen Benutzennamen aus dem Namensset.

Es wird vorausgesetzt, dass der Name im Set vorhanden ist.

Parameter

<i>name</i>	ist der Name der gelöscht wird
-------------	--------------------------------

3.71.2.3 void addGameServer (GameServer game)

Fügt einen neuen [GameServer](#) in das gameServerSet ein.

Parameter

<i>game</i>	ist der GameServer der eingefügt wird
-------------	---

3.71.2.4 void removeGameServer ([GameServer](#) *game*)

Löscht einen [GameServer](#) aus dem Gameserverset.

Parameter

<i>game</i>	ist der GameServer der gelöscht wird
-------------	--

3.71.2.5 void receiveMessage ([Player](#) *player*, [ComChatMessage](#) *chat*)

Diese überladene Methode ist dafür zuständig eine Chatnachricht an alle Clients im Spiel zu verschicken.

Dafür wird die [ComChatMessage](#) mit broadcast an alle Spieler im [playerSet](#) verteilt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>chat</i>	ist das ComObject , das die Chatnachricht enthält

Benutzt [Server.broadcast\(\)](#).

3.71.2.6 void receiveMessage ([Player](#) *player*, [ComClientQuit](#) *quit*)

Diese überladene Methode schließt die Verbindung, der [Player](#) wird aus dem [playerSet](#) (bzw.

[noNames Set](#)) entfernt, der Name des Players wird aus dem [Set names](#) entfernt. War der Spieler im [playerSet](#), wird ein [ComUpdatePlayerlist](#) mit broadcast an alle Clients verschickt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>quit</i>	ist das ComObject , welches angibt, dass der Spieler das Spiel vollständig verlässt

3.71.2.7 void receiveMessage ([Player](#) *player*, [ComCreateGameRequest](#) *create*)

Diese überladene Methode erstellt einen neuen [GameServer](#) fügt ihm den [Player](#) durch aufruf von dessen [changeServer](#) Methode hinzu.

Der neue [GameServer](#) wird in das [gameServerSet](#) eingefügt. Durch broadcast wird im [LobbyServer](#) sowohl [ComUpdatePlayerlist](#) als auch ein [ComLobbyUpdateGamelist](#) verschickt. Zusätzlich wird dem Client mit [sendToPlayer](#) ein [ComInitGameLobby](#) geschickt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>create</i>	ist das ComObject , welches angibt, dass der Player ein neues Spiel erstellt hat

3.71.2.8 void receiveMessage ([Player](#) *player*, [ComJoinRequest](#) *join*)

Diese überladene Methode fügt einen [Player](#) dem entsprechenden [GameServer](#) hinzu.

Falls das Passwort nicht leer ist wird geprüft, ob es mit dem Passwort des Spieles übereinstimmt, wenn nicht, wird ein [ComWarning](#) an den Client geschickt. Ansonsten wird und der [Player](#) dem, durch Namen des Spielleiters identifizierten, durch Aufruf von [changeServer](#) Gameserver übergeben. Dem joinendenClient wird mit [sendToPlayer](#) ein [ComInitGameLobby](#) geschickt. Durch broadcast wird sowohl im [LobbyServer](#) ein [ComUpdatePlayerlist](#) verschickt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>join</i>	ist das ComObject, welches angibt, dass der Player einem Spiel beitreten will

3.71.2.9 void receiveMessage ([Player](#) *player*, ComLoginRequest *login*)

Diese überladene Methode überprüft, ob der Name im Set names vorhanden ist, falls ja, wird ein ComWarning an den Client geschickt, dass der Name bereits vergeben ist, falls nein, wird im [Player](#) setName aufgerufen.

Der [Player](#) wird aus dem noNames Set entfernt und in das playerSet eingefügt. Der Name wird in das Set names eingefügt. Dem Client wird ein ComServerAcknowledgement geschickt.

Parameter

<i>player</i>	ist der Thread der die Nachricht erhalten hat
<i>login</i>	ist das ComObject, dass den Benutzernamen des Clients enthält

3.71.2.10 ComInitLobby initLobby ()

Diese Methode baut ein neues ComInitLobby Objekt und gibt es zurück.

Rückgabe

Gibt das ComInitLobby Objekt zurück

3.71.2.11 void handleIOException ([Player](#) *player*)

Diese Methode legt den Ablauf fest, was passiert, falls die Verbindung zu einem Client verloren gegangen ist.

Der übergebene [Player](#) wird aus dem playerSet sowie dem names Set im [LobbyServer](#) entfernt.

Parameter

<i>player</i>	ist der Tread von dem die IOException kommt
---------------	---

3.72 LobbyServer.ClientListenerThread Klassenreferenz

Abgeleitet von Runnable.

Öffentliche Methoden

- void [run](#) ()

3.72.1 Ausführliche Beschreibung

Der Thread auf eingehende Clientverbindungen, stellt diese her und instanziiert für jede Verbindung eine Klasse [Player](#). Dieser wird dann dem [LobbyServer](#) übergeben.

Autor

Viktoria

3.73 Player Klassenreferenz

Abgeleitet von Runnable.

Öffentliche Methoden

- [Player](#) ([Server](#) lobbyServer, ObjectOutputStream output, ObjectInputStream input)
- void [run](#) ()
- void [send](#) (ComObject com)
- void [changeServer](#) ([Server](#) newServer)
- String [getName](#) ()
- void [setName](#) (String newName)

Private Attribute

- String [name](#)
- [Server](#) [server](#)
- ObjectOutputStream [comOut](#)
- ObjectInputStream [comIn](#)

3.73.1 Ausführliche Beschreibung

Sie verwaltet für die Dauer einer Serververbindung die Verbindung zu einem Client.

3.73.2 Beschreibung der Konstruktoren und Destruktoren

3.73.2.1 [Player](#) ([Server](#) *lobbyServer*, ObjectOutputStream *output*, ObjectInputStream *input*)

Konstruktor des [Players](#), in ihm werden die Attribute [server](#), [comOut](#) und [comIn](#) mit vom [ClientListenerThread](#) übergebenen Werten instanziiert.

Parameter

<i>lobbyServer</i>	ist der LobbyServer , der zu Beginn den Player verwaltet.
<i>output</i>	ist der ObjectOutputStream an den entsprechenden Client
<i>input</i>	ist der ObjectInputStream vom entsprechenden Client

Benutzt [Player.comIn](#), [Player.comOut](#) und [Player.server](#).

3.73.3 Dokumentation der Elementfunktionen

3.73.3.1 void [run](#) ()

Die [run](#)-Methode des Thread nimmt eingehende Nachrichten des Client entgegen und übergibt diese an den [Server](#) durch Aufruf der Methode [resolveMessage\(\)](#). Fängt eine [ClassNotFoundException](#) ab, falls die Klasse nicht gefunden werden kann und gibt einen Fehler aus.

Fängt eine [IOException](#) ab und ruft im jeweiligen [Server](#), dem er zugeteilt ist die [handleIOException](#) Methode auf.

Benutzt [Player.comIn](#) und [Player.server](#).

3.73.3.2 void [send](#) (ComObject *com*)

Diese Methode schickt ein [ComObject](#) an den Client.

Sie fängt eine [IOException](#) ab und ruft im jeweiligen [Server](#), dem er zugeteilt ist die [handleIOException](#) Methode auf.

Parameter

<i>com</i>	ist das ComObject das verschickt wird
------------	---------------------------------------

3.73.3.3 void changeServer ([Server](#) *newServer*)

Diese Methode wechselt beim [Player](#) den [Server](#) an den er comObjects weiterleiten soll.

Dabei wird er aus dem playerSet des alten Servers entfernt und in das playerSet des neuen Players eingefügt. Danach wird vom neuen [Server](#) ein ComUpdatePlayerlist Objekt mit broadcast an alle Clients, die vom [Server](#) verwaltet werden, verschickt.

Parameter

<i>newServer</i>	ist der neue Server
------------------	-------------------------------------

Benutzt Player.getName() und Player.server.

3.73.3.4 String getName ()

Getter-Methode für den Benutzernamen.

Rückgabe

gibt den Benutzernamen des Spielers zurück

Benutzt Player.name.

Wird benutzt von Player.changeServer().

3.73.3.5 void setName ([String](#) *newName*)

Setter-Methode für den Benutzernamen.

Parameter

<i>newName</i>	ist der neue Name
----------------	-------------------

Benutzt Player.name.

3.74 Server Klassenreferenz

Basisklasse für [GameServer](#) und [LobbyServer](#).

Öffentliche Methoden

- void [receiveMessage](#) ([Player](#) player, ComObject com)
- synchronized void [sendToPlayer](#) (String name, ComObject com)
- synchronized void [addPlayer](#) ([Player](#) player)
- synchronized void [removePlayer](#) ([Player](#) player)
- synchronized void [broadcast](#) (ComObject com)
- void [handleIOException](#) ([Player](#) player)

Geschützte Attribute

- Set< [Player](#) > [playerSet](#)

3.74.1 Ausführliche Beschreibung

Es stellt Methoden zur Nachrichtenversendung und -verarbeitung bereit, sowie zur Verwaltung von Playern

3.74.2 Dokumentation der Elementfunktionen

3.74.2.1 void receiveMessage (Player player, ComObject com)

Diese Methode dient zur Verarbeitung von eingehenden ComObjects.

Parameter

<i>player</i>	ist der Player von dem die Nachricht kommt
<i>com</i>	ist das ComObject vom Client verschickt wurde

3.74.2.2 synchronized void sendToPlayer (String name, ComObject com)

Diese Methode wird genutzt, um ein ComObject an einen einzigen Client zu verschicken.

Der [Player](#) der die Nachricht verschicken soll wird Anhand des übergebenen Benutzernamens identifiziert. Es wird vorausgesetzt, dass der Name und das ComObject gültig sind.

Parameter

<i>name</i>	ist der Name des Clients, an den der Player die Nachricht verschicken soll
<i>c</i>	ist das ComObject, dass verschickt werden soll

Benutzt Server.playerSet.

3.74.2.3 synchronized void addPlayer (Player player)

Diese Methode fügt einen [Player](#) dem Set an Playern hinzu, welche der [Server](#) verwaltet.

Es wird vorausgesetzt, dass der [Player](#) gültig und noch nicht im Set vorhanden ist.

Parameter

<i>player</i>	ist der Player , der hinzugefügt wird
---------------	---

3.74.2.4 synchronized void removePlayer (Player player)

Diese Methode entfernt einen [Player](#) aus dem Set an Playern, welche der [Server](#) verwaltet.

Es wird vorausgesetzt, dass der [Player](#) gültig und im Set vorhanden ist.

Parameter

<i>player</i>	ist der Player , der entfernt wird
---------------	--

3.74.2.5 synchronized void broadcast (ComObject com)

Diese Methode wird genutzt, um ein ComObject an alle Clients, die vom [Server](#) verwaltet werden, zu schicken.

Es wird vorausgesetzt, dass das ComObject gültig ist.

Parameter

<i>com</i>	ist das ComObject, dass verschickt werden soll
------------	--

Benutzt Server.playerSet.

Wird benutzt von LobbyServer.receiveMessage() und GameServer.receiveMessage().

3.74.2.6 void handleIOException (Player player)

Diese Methode legt den Ablauf fest, was passiert, falls die Verbindung zu einem Client verloren gegangen ist.

Parameter

<i>player</i>	ist der Tread von dem die IOException kommt
---------------	---

3.75 ServerMain Klassenreferenz

Öffentliche, statische Methoden

- static void [main](#) (String[] args)

Private Attribute

- [LobbyServer lobbyServer](#)

3.75.1 Dokumentation der Elementfunktionen

3.75.1.1 static void main (String[] args) [static]

Die main-Methode erstellt einen neuen [LobbyServer](#).

Parameter

<i>args</i>	
-------------	--

Index

- addCard
 - Ruleset::PlayerState, 65
- addChatMessageListener
 - Client::View::GameLobby, 21
 - Client::View::Lobby, 26
- addConnectButtonListener
 - Client::View::Login, 27
- addCreateButtonListener
 - Client::View::CreateGame, 18
- addGameServer
 - Server::LobbyServer, 81
- addHostButtonListener
 - Client::View::Lobby, 25
- addJoinButtonListener
 - Client::View::Lobby, 25
 - Client::View::Password, 29
- addLanguageSelectionListener
 - Client::View::Login, 27
- addLeaveButtonListener
 - Client::View::CreateGame, 18
 - Client::View::GameLobby, 21
 - Client::View::Lobby, 25
- addName
 - Server::LobbyServer, 81
- addPanelMouseListener
 - Client::View::CreateGame, 18
- addPlayer
 - Server::GameServer, 77
 - Server::Server, 86
- addPlayerToGame
 - Ruleset::GameState, 59
 - Ruleset::ServerRuleset, 71
- addRemoveButtonListener
 - Client::View::GameLobby, 21
- addRulesetSelectionListener
 - Client::View::CreateGame, 18
- addStartButtonListener
 - Client::View::GameLobby, 21
- broadcast
 - Ruleset::ServerRuleset, 72
 - Server::Server, 86
- Card, 15, 48
 - Client::View::Card, 15
- changeServer
 - Server::Player, 85
- ChooseCards, 16
- ChooseItem, 16
- Client::ClientMain
 - main, 8
- Client::ClientModel
 - createConnection, 14
 - getLanguage, 12
 - getLobbyGamelist, 11
 - getOtherPlayerData, 12
 - getOwnHand, 11
 - getOwnScore, 12
 - getPlayedCards, 11
 - getPlayerCount, 13
 - getPlayerlist, 11
 - getRulesets, 14
 - getWarningText, 14
 - getWindowText, 13
 - hostGame, 12
 - informView, 14
 - initGame, 14
 - joinGame, 13
 - kickPlayer, 12
 - leaveWindow, 9
 - makeMove, 14
 - receiveMessage, 9–11
 - send, 13
 - sendChatMessage, 13
 - sendMessage, 13
 - setLanguage, 12
 - startGame, 14
- Client::View::Card
 - Card, 15
 - getID, 16
- Client::View::ChooseCards
 - update, 16
- Client::View::ChooseItem
 - update, 17
- Client::View::CreateGame
 - addCreateButtonListener, 18
 - addLeaveButtonListener, 18
 - addPanelMouseListener, 18
 - addRulesetSelectionListener, 18
 - setLanguage, 18
- Client::View::Game
 - Game, 19
 - makeTrickGameBoard, 19
 - update, 20
- Client::View::GameLobby
 - addChatMessageListener, 21
 - addLeaveButtonListener, 21
 - addRemoveButtonListener, 21
 - addStartButtonListener, 21
 - setLanguage, 21
 - update, 21, 23
- Client::View::GamePanel
 - setupTrickGame, 23
- Client::View::InputNumber
 - update, 24
- Client::View::Lobby
 - addChatMessageListener, 26
 - addHostButtonListener, 25
 - addJoinButtonListener, 25
 - addLeaveButtonListener, 25
 - setLanguage, 26

- update, 26
- Client::View::Login
 - addConnectButtonListener, 27
 - addLanguageSelectionListener, 27
 - setLanguage, 27
 - update, 27
- Client::View::Password
 - addJoinButtonListener, 29
 - setLanguage, 29
 - update, 29
- Client::View::ScoreWindow
 - update, 29
- Client::View::ViewCard
 - getID, 30
 - ViewCard, 30
- Client::View::Warning
 - update, 31
- ClientHearts, 49
 - Ruleset::ClientHearts, 49
- ClientMain, 8
- ClientModel, 8
- ClientRuleset, 50
 - Ruleset::ClientRuleset, 50
- ClientState, 15
- ClientWizard, 53
 - Ruleset::ClientWizard, 53
- Colour, 54
- ComBeenKicked, 31
 - ComObjects::ComBeenKicked, 31
- ComChatMessage, 32
 - ComObjects::ComChatMessage, 32
- ComClientLeave, 32
- ComClientQuit, 33
- ComCreateGameRequest, 33
 - ComObjects::ComCreateGameRequest, 33
- ComInitGameLobby, 34
 - ComObjects::ComInitGameLobby, 35
- ComInitLobby, 35
 - ComObjects::ComInitLobby, 35
- ComJoinRequest, 36
 - ComObjects::ComJoinRequest, 36
- ComKickPlayerRequest, 37
 - ComObjects::ComKickPlayerRequest, 37
- ComLobbyUpdateGamelist, 38
 - ComObjects::ComLobbyUpdateGamelist, 38
- ComLoginRequest, 39
 - ComObjects::ComLoginRequest, 39
- ComObjects::ComBeenKicked
 - ComBeenKicked, 31
 - getMessage, 32
- ComObjects::ComChatMessage
 - ComChatMessage, 32
 - getChatMessage, 32
- ComObjects::ComCreateGameRequest
 - ComCreateGameRequest, 33
 - getGameName, 34
 - getPassword, 34
 - getRuleset, 34
- ComObjects::ComInitGameLobby
 - hasPassword, 34
- ComObjects::ComInitGameLobby
 - ComInitGameLobby, 35
 - getPlayerList, 35
- ComObjects::ComInitLobby
 - ComInitLobby, 35
 - getGameList, 36
 - getPlayerList, 36
- ComObjects::ComJoinRequest
 - ComJoinRequest, 36
 - gameMasterName, 37
 - getGameMasterName, 37
- ComObjects::ComKickPlayerRequest
 - ComKickPlayerRequest, 37
 - getPlayerName, 38
- ComObjects::ComLobbyUpdateGamelist
 - ComLobbyUpdateGamelist, 38
 - getGameServer, 39
 - isRemoveFlag, 38
- ComObjects::ComLoginRequest
 - ComLoginRequest, 39
 - getPlayerName, 39
- ComObjects::ComRuleset
 - ComRuleset, 40
 - getRulesetMessage, 40
- ComObjects::ComUpdatePlayerlist
 - ComUpdatePlayerlist, 41
 - getPlayerName, 41
 - isRemoveFlag, 41
- ComObjects::ComWarning
 - ComWarning, 42
 - getWarning, 42
- ComObjects::MsgCard
 - getCard, 43
 - MsgCard, 43
- ComObjects::MsgMultiCards
 - getCardList, 44
 - MsgMultiCards, 44
- ComObjects::MsgMultiCardsRequest
 - getCount, 44
- ComObjects::MsgNumber
 - getNumber, 45
 - MsgNumber, 45
- ComObjects::MsgNumberRequest
 - MsgNumberRequest, 46
- ComObjects::MsgSelection
 - getSelection, 46
 - MsgSelection, 46
- ComObjects::MsgUser
 - getGameClientUpdate, 47
 - MsgUser, 47
- ComObjects::RulesetMessage
 - visit, 48
- ComRuleset, 39
 - ComObjects::ComRuleset, 40
- ComServerAcknowledgement, 40
- ComStartGame, 40
- ComUpdatePlayerlist, 41

- ComObjects::ComUpdatePlayerlist, 41
- ComWarning, 41
 - ComObjects::ComWarning, 42
- createConnection
 - Client::ClientModel, 14
- createDeck
 - Ruleset::ServerRuleset, 69
- CreateGame, 17
- dealCards
 - Ruleset::GameState, 61
 - Ruleset::ServerRuleset, 72
- DiscardPile, 18
- DrawDeck, 18
- Game, 19
 - Client::View::Game, 19
- GameClientUpdate, 55
 - Ruleset::GameClientUpdate, 55
- GameLobby, 20
- gameMasterName
 - ComObjects::ComJoinRequest, 37
- GamePanel, 23
- GamePhase, 57
- GameServer, 76
 - Server::GameServer, 77
- GameServerRepresentation, 79
 - Server::GameServerRepresentation, 80
- GameState, 57
 - Ruleset::GameState, 58
- getAnnouncedTricks
 - Ruleset::WizData, 76
- getCard
 - ComObjects::MsgCard, 43
- getCardList
 - ComObjects::MsgMultiCards, 44
- getCardsLeftInDeck
 - Ruleset::GameState, 59
- getChatMessage
 - ComObjects::ComChatMessage, 32
- getColour
 - Ruleset::Card, 48
 - Ruleset::HeartsCard, 62
 - Ruleset::WizardCard, 75
- getCount
 - ComObjects::MsgMultiCardsRequest, 44
- getCurrentPlayer
 - Ruleset::ClientRuleset, 52
 - Ruleset::GameClientUpdate, 56
 - Ruleset::GameState, 59
 - Ruleset::ServerRuleset, 71
- getFirstPlayer
 - Ruleset::GameState, 59
 - Ruleset::ServerRuleset, 71
- getGameClientUpdate
 - ComObjects::MsgUser, 47
- getGameList
 - ComObjects::ComInitLobby, 36
- getGameMasterName
 - ComObjects::ComJoinRequest, 37
- getGameName
 - ComObjects::ComCreateGameRequest, 34
- getGamePhase
 - Ruleset::ClientRuleset, 51
 - Ruleset::ServerRuleset, 69
- getGameServer
 - ComObjects::ComLobbyUpdateGamelist, 39
- getHand
 - Ruleset::PlayerState, 65
- getID
 - Client::View::Card, 16
 - Client::View::ViewCard, 30
- getLanguage
 - Client::ClientModel, 12
- getLobbyGamelist
 - Client::ClientModel, 11
- getMaxPlayers
 - Ruleset::ClientRuleset, 51
 - Ruleset::ServerRuleset, 69
- getMessage
 - ComObjects::ComBeenKicked, 32
- getMinPlayers
 - Ruleset::ClientRuleset, 51
 - Ruleset::ServerRuleset, 69
- getName
 - Ruleset::PlayerState, 65
 - Server::Player, 85
- getNumber
 - ComObjects::MsgNumber, 45
- getNumberOfPlayedCards
 - Ruleset::GameState, 60
- getNumberOfTricks
 - Ruleset::OtherData, 63
- getOtherData
 - Ruleset::PlayerState, 65
 - Ruleset::ServerRuleset, 71
- getOtherPlayerData
 - Client::ClientModel, 12
 - Ruleset::ClientRuleset, 52
 - Ruleset::GameClientUpdate, 56
- getOwnData
 - Ruleset::ClientRuleset, 51
 - Ruleset::GameClientUpdate, 56
- getOwnHand
 - Client::ClientModel, 11
 - Ruleset::ClientRuleset, 51
 - Ruleset::GameClientUpdate, 56
- getOwnScore
 - Client::ClientModel, 12
- getPassword
 - ComObjects::ComCreateGameRequest, 34
- getPlayedCards
 - Client::ClientModel, 11
 - Ruleset::GameClientUpdate, 56
 - Ruleset::GameState, 60
- getPlayerCards
 - Ruleset::GameState, 61

- Ruleset::ServerRuleset, 72
- getPlayerCount
 - Client::ClientModel, 13
- getPlayerList
 - ComObjects::ComInitGameLobby, 35
 - ComObjects::ComInitLobby, 36
- getPlayerName
 - ComObjects::ComKickPlayerRequest, 38
 - ComObjects::ComLoginRequest, 39
 - ComObjects::ComUpdatePlayerlist, 41
- getPlayerState
 - Ruleset::GameState, 60
 - Ruleset::ServerRuleset, 72
- getPlayerlist
 - Client::ClientModel, 11
- getPoints
 - Ruleset::OtherData, 64
- getRoundNumber
 - Ruleset::GameState, 60
- getRuleset
 - ComObjects::ComCreateGameRequest, 34
- getRulesetMessage
 - ComObjects::ComRuleset, 40
- getRulesetType
 - Ruleset::ClientRuleset, 51
 - Ruleset::ServerRuleset, 69
- getRulesets
 - Client::ClientModel, 14
- getSelection
 - ComObjects::MsgSelection, 46
- getTrumpCard
 - Ruleset::ClientRuleset, 52
 - Ruleset::GameClientUpdate, 56
 - Ruleset::GameState, 60
- getValue
 - Ruleset::Card, 48
 - Ruleset::HeartsCard, 62
 - Ruleset::WizardCard, 75
- getWarning
 - ComObjects::ComWarning, 42
- getWarningText
 - Client::ClientModel, 14
- getWindowText
 - Client::ClientModel, 13
- giveACard
 - Ruleset::GameState, 61
 - Ruleset::ServerRuleset, 73
- handleIOException
 - Server::GameServer, 79
 - Server::LobbyServer, 83
 - Server::Server, 86
- hasPassword
 - ComObjects::ComCreateGameRequest, 34
- HeartsCard, 62
- HeartsData, 62
- hostGame
 - Client::ClientModel, 12
- informView
 - Client::ClientModel, 14
- initGame
 - Client::ClientModel, 14
- initLobby
 - Server::GameServer, 79
 - Server::LobbyServer, 83
- InputNumber, 24
- isRemoveFlag
 - ComObjects::ComLobbyUpdateGamelist, 38
 - ComObjects::ComUpdatePlayerlist, 41
- isValidMove
 - Ruleset::ClientHearts, 49
 - Ruleset::ClientRuleset, 53
 - Ruleset::ClientWizard, 54
 - Ruleset::ServerHearts, 66
 - Ruleset::ServerRuleset, 73
 - Ruleset::ServerWizard, 74
- joinGame
 - Client::ClientModel, 13
- kickPlayer
 - Client::ClientModel, 12
- Language, 24
- leaveWindow
 - Client::ClientModel, 9
- Lobby, 24
- LobbyServer, 80
- LobbyServer.ClientListenerThread, 83
- Login, 26
- madeTrick
 - Ruleset::OtherData, 63
- main
 - Client::ClientMain, 8
 - Server::ServerMain, 87
- makeMove
 - Client::ClientModel, 14
- makeTrickGameBoard
 - Client::View::Game, 19
- MsgCard, 42
 - ComObjects::MsgCard, 43
- MsgCardRequest, 43
- MsgGameEnd, 43
- MsgMultiCards, 43
 - ComObjects::MsgMultiCards, 44
- MsgMultiCardsRequest, 44
- MsgMultipleCardsRequest, 45
- MsgNumber, 45
 - ComObjects::MsgNumber, 45
- MsgNumberRequest, 45
 - ComObjects::MsgNumberRequest, 46
- MsgSelection, 46
 - ComObjects::MsgSelection, 46
- MsgSelectionRequest, 46
- MsgUser, 47
 - ComObjects::MsgUser, 47

- nextPlayer
 - Ruleset::ServerRuleset, 71
- OtherData, 63
- OtherPlayer, 28
- OwnHand, 28
- Password, 28
- password
 - Server::GameServer, 79
- playCard
 - Ruleset::GameState, 61
 - Ruleset::ServerRuleset, 73
- Player, 83
 - Server::Player, 84
- PlayerState, 64
 - Ruleset::PlayerState, 64
- receiveMessage
 - Client::ClientModel, 9–11
 - Server::GameServer, 78, 79
 - Server::LobbyServer, 82, 83
 - Server::Server, 86
- removeCard
 - Ruleset::PlayerState, 65
- removeGameServer
 - Server::LobbyServer, 82
- removeName
 - Server::LobbyServer, 81
- removePlayer
 - Server::GameServer, 77
 - Server::Server, 86
- removeTricks
 - Ruleset::OtherData, 63
- resolveMessage
 - Ruleset::ClientHearts, 49
 - Ruleset::ClientRuleset, 52
 - Ruleset::ClientWizard, 54
 - Ruleset::ServerHearts, 66
 - Ruleset::ServerRuleset, 72
 - Ruleset::ServerWizard, 74
- Ruleset::Card
 - getColour, 48
 - getValue, 48
- Ruleset::ClientHearts
 - ClientHearts, 49
 - isValidMove, 49
 - resolveMessage, 49
 - send, 49
- Ruleset::ClientRuleset
 - ClientRuleset, 50
 - getCurrentPlayer, 52
 - getGamePhase, 51
 - getMaxPlayers, 51
 - getMinPlayers, 51
 - getOtherPlayerData, 52
 - getOwnData, 51
 - getOwnHand, 51
 - getRulesetType, 51
 - getTrumpCard, 52
 - isValidMove, 53
 - resolveMessage, 52
 - send, 53
- Ruleset::ClientWizard
 - ClientWizard, 53
 - isValidMove, 54
 - resolveMessage, 54
 - send, 54
- Ruleset::GameClientUpdate
 - GameClientUpdate, 55
 - getCurrentPlayer, 56
 - getOtherPlayerData, 56
 - getOwnData, 56
 - getOwnHand, 56
 - getPlayedCards, 56
 - getTrumpCard, 56
- Ruleset::GameState
 - addPlayerToGame, 59
 - dealCards, 61
 - GameState, 58
 - getCardsLeftInDeck, 59
 - getCurrentPlayer, 59
 - getFirstPlayer, 59
 - getNumberOfPlayedCards, 60
 - getPlayedCards, 60
 - getPlayerCards, 61
 - getPlayerState, 60
 - getRoundNumber, 60
 - getTrumpCard, 60
 - giveACard, 61
 - playCard, 61
 - setCurrentPlayer, 59
 - setFirstPlayer, 59
 - setTrumpCard, 60
- Ruleset::HeartsCard
 - getColour, 62
 - getValue, 62
- Ruleset::OtherData
 - getNumberOfTricks, 63
 - getPoints, 64
 - madeTrick, 63
 - removeTricks, 63
 - setPoints, 63
- Ruleset::PlayerState
 - addCard, 65
 - getHand, 65
 - getName, 65
 - getOtherData, 65
 - PlayerState, 64
 - removeCard, 65
- Ruleset::ServerHearts
 - isValidMove, 66
 - resolveMessage, 66
- Ruleset::ServerRuleset
 - addPlayerToGame, 71
 - broadcast, 72
 - createDeck, 69

- dealCards, 72
- getCurrentPlayer, 71
- getFirstPlayer, 71
- getGamePhase, 69
- getMaxPlayers, 69
- getMinPlayers, 69
- getOtherData, 71
- getPlayerCards, 72
- getPlayerState, 72
- getRulesetType, 69
- giveACard, 73
- isValidMove, 73
- nextPlayer, 71
- playCard, 73
- resolveMessage, 72
- send, 72
- ServerRuleset, 68
- setCurrentPlayer, 71
- setFirstPlayer, 69
- setTrumpCard, 73
- Ruleset::ServerWizard
 - isValidMove, 74
 - resolveMessage, 74
- Ruleset::WizData
 - getAnnouncedTricks, 76
 - setAnnounceTricks, 76
- Ruleset::WizardCard
 - getColour, 75
 - getValue, 75
- RulesetMessage, 47
- RulesetType, 65
- run
 - Server::Player, 84
- ScoreWindow, 29
- send
 - Client::ClientModel, 13
 - Ruleset::ClientHearts, 49
 - Ruleset::ClientRuleset, 53
 - Ruleset::ClientWizard, 54
 - Ruleset::ServerRuleset, 72
 - Server::Player, 84
- sendChatMessage
 - Client::ClientModel, 13
- sendMessage
 - Client::ClientModel, 13
- sendToPlayer
 - Server::Server, 86
- Server, 85
- Server::GameServer
 - addPlayer, 77
 - GameServer, 77
 - handleIOException, 79
 - initLobby, 79
 - password, 79
 - receiveMessage, 78, 79
 - removePlayer, 77
- Server::GameServerRepresentation
 - GameServerRepresentation, 80
- Server::LobbyServer
 - addGameServer, 81
 - addName, 81
 - handleIOException, 83
 - initLobby, 83
 - receiveMessage, 82, 83
 - removeGameServer, 82
 - removeName, 81
- Server::Player
 - changeServer, 85
 - getName, 85
 - Player, 84
 - run, 84
 - send, 84
 - setName, 85
- Server::Server
 - addPlayer, 86
 - broadcast, 86
 - handleIOException, 86
 - receiveMessage, 86
 - removePlayer, 86
 - sendToPlayer, 86
- Server::ServerMain
 - main, 87
- ServerHearts, 65
- ServerMain, 87
- ServerRuleset, 66
 - Ruleset::ServerRuleset, 68
- ServerWizard, 73
- setAnnounceTricks
 - Ruleset::WizData, 76
- setCurrentPlayer
 - Ruleset::GameState, 59
 - Ruleset::ServerRuleset, 71
- setFirstPlayer
 - Ruleset::GameState, 59
 - Ruleset::ServerRuleset, 69
- setLanguage
 - Client::ClientModel, 12
 - Client::View::CreateGame, 18
 - Client::View::GameLobby, 21
 - Client::View::Lobby, 26
 - Client::View::Login, 27
 - Client::View::Password, 29
- setName
 - Server::Player, 85
- setPoints
 - Ruleset::OtherData, 63
- setTrumpCard
 - Ruleset::GameState, 60
 - Ruleset::ServerRuleset, 73
- setupTrickGame
 - Client::View::GamePanel, 23
- startGame
 - Client::ClientModel, 14
- update
 - Client::View::ChooseCards, 16
 - Client::View::ChooseItem, 17

- Client::View::Game, [20](#)
- Client::View::GameLobby, [21](#), [23](#)
- Client::View::InputNumber, [24](#)
- Client::View::Lobby, [26](#)
- Client::View::Login, [27](#)
- Client::View::Password, [29](#)
- Client::View::ScoreWindow, [29](#)
- Client::View::Warning, [31](#)
- ViewCard, [30](#)
 - Client::View::ViewCard, [30](#)
- visit
 - ComObjects::RulesetMessage, [48](#)
- Warning, [31](#)
- WizData, [76](#)
- WizardCard, [75](#)