

NET-WizHearts

Erzeugt von Doxygen 1.8.5

Mon Nov 11 2013 23:15:20

## Inhaltsverzeichnis

<b>1</b>	<b>Hierarchie-Verzeichnis</b>	<b>1</b>
1.1	Klassenhierarchie . . . . .	1
<b>2</b>	<b>Klassen-Verzeichnis</b>	<b>4</b>
2.1	Auflistung der Klassen . . . . .	4
<b>3</b>	<b>Klassen-Dokumentation</b>	<b>8</b>
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 . . . . .	12
3.4	Card Klassenreferenz . . . . .	13
3.4.1	Ausführliche Beschreibung . . . . .	13
3.4.2	Beschreibung der Konstruktoren und Destruktoren . . . . .	13
3.4.3	Dokumentation der Elementfunktionen . . . . .	13
3.5	ChooseCards Klassenreferenz . . . . .	14
3.5.1	Dokumentation der Elementfunktionen . . . . .	14
3.6	Chooseltem Klassenreferenz . . . . .	14
3.6.1	Dokumentation der Elementfunktionen . . . . .	14
3.7	CreateGame Klassenreferenz . . . . .	14
3.7.1	Ausführliche Beschreibung . . . . .	15
3.7.2	Dokumentation der Elementfunktionen . . . . .	15
3.8	DiscardPile Klassenreferenz . . . . .	16
3.9	DrawDeck Klassenreferenz . . . . .	16
3.10	Game Klassenreferenz . . . . .	16
3.10.1	Ausführliche Beschreibung . . . . .	17
3.10.2	Beschreibung der Konstruktoren und Destruktoren . . . . .	17
3.10.3	Dokumentation der Elementfunktionen . . . . .	17
3.11	GameLobby Klassenreferenz . . . . .	17
3.11.1	Ausführliche Beschreibung . . . . .	18
3.11.2	Dokumentation der Elementfunktionen . . . . .	18
3.12	GamePanel Klassenreferenz . . . . .	19
3.12.1	Ausführliche Beschreibung . . . . .	20
3.12.2	Dokumentation der Elementfunktionen . . . . .	20
3.13	InputNumber Klassenreferenz . . . . .	20
3.13.1	Dokumentation der Elementfunktionen . . . . .	20
3.14	Language Enum-Referenz . . . . .	21

3.15	Lobby Klassenreferenz	21
3.15.1	Ausführliche Beschreibung	22
3.15.2	Dokumentation der Elementfunktionen	22
3.16	Login Klassenreferenz	23
3.16.1	Ausführliche Beschreibung	24
3.16.2	Dokumentation der Elementfunktionen	24
3.17	OtherPlayer Klassenreferenz	24
3.18	OwnHand Klassenreferenz	24
3.18.1	Ausführliche Beschreibung	25
3.19	Password Klassenreferenz	25
3.19.1	Dokumentation der Elementfunktionen	25
3.20	ScoreWindow Klassenreferenz	26
3.20.1	Dokumentation der Elementfunktionen	26
3.21	ViewCard Klassenreferenz	26
3.21.1	Ausführliche Beschreibung	27
3.21.2	Beschreibung der Konstruktoren und Destruktoren	27
3.21.3	Dokumentation der Elementfunktionen	27
3.22	Warning Klassenreferenz	27
3.22.1	Ausführliche Beschreibung	27
3.22.2	Dokumentation der Elementfunktionen	27
3.23	ComBeenKicked Klassenreferenz	29
3.23.1	Ausführliche Beschreibung	29
3.23.2	Beschreibung der Konstruktoren und Destruktoren	29
3.23.3	Dokumentation der Elementfunktionen	29
3.24	ComChatMessage Klassenreferenz	29
3.24.1	Ausführliche Beschreibung	30
3.24.2	Beschreibung der Konstruktoren und Destruktoren	30
3.24.3	Dokumentation der Elementfunktionen	30
3.25	ComClientLeave Klassenreferenz	30
3.25.1	Ausführliche Beschreibung	30
3.26	ComClientQuit Klassenreferenz	30
3.26.1	Ausführliche Beschreibung	30
3.27	ComCreateGameRequest Klassenreferenz	30
3.27.1	Ausführliche Beschreibung	31
3.27.2	Beschreibung der Konstruktoren und Destruktoren	31
3.27.3	Dokumentation der Elementfunktionen	31
3.28	ComInitGameLobby Klassenreferenz	32
3.28.1	Ausführliche Beschreibung	32
3.28.2	Beschreibung der Konstruktoren und Destruktoren	32
3.28.3	Dokumentation der Elementfunktionen	32

3.29	ComInitLobby Klassenreferenz	33
3.29.1	Ausführliche Beschreibung	33
3.29.2	Beschreibung der Konstruktoren und Destruktoren	33
3.29.3	Dokumentation der Elementfunktionen	33
3.30	ComJoinRequest Klassenreferenz	33
3.30.1	Ausführliche Beschreibung	34
3.30.2	Beschreibung der Konstruktoren und Destruktoren	34
3.30.3	Dokumentation der Elementfunktionen	34
3.30.4	Dokumentation der Datenelemente	34
3.31	ComKickPlayerRequest Klassenreferenz	35
3.31.1	Ausführliche Beschreibung	35
3.31.2	Beschreibung der Konstruktoren und Destruktoren	35
3.31.3	Dokumentation der Elementfunktionen	35
3.32	ComLobbyUpdateGamelist Klassenreferenz	35
3.32.1	Ausführliche Beschreibung	36
3.32.2	Beschreibung der Konstruktoren und Destruktoren	36
3.32.3	Dokumentation der Elementfunktionen	36
3.33	ComLoginRequest Klassenreferenz	36
3.33.1	Ausführliche Beschreibung	36
3.33.2	Beschreibung der Konstruktoren und Destruktoren	37
3.33.3	Dokumentation der Elementfunktionen	37
3.34	ComRuleset Klassenreferenz	37
3.34.1	Ausführliche Beschreibung	37
3.34.2	Beschreibung der Konstruktoren und Destruktoren	37
3.34.3	Dokumentation der Elementfunktionen	37
3.35	ComServerAcknowledgement Klassenreferenz	38
3.36	ComStartGame Klassenreferenz	38
3.36.1	Ausführliche Beschreibung	38
3.37	ComUpdatePlayerlist Klassenreferenz	38
3.37.1	Ausführliche Beschreibung	38
3.37.2	Beschreibung der Konstruktoren und Destruktoren	38
3.37.3	Dokumentation der Elementfunktionen	39
3.38	ComWarning Klassenreferenz	39
3.38.1	Ausführliche Beschreibung	39
3.38.2	Beschreibung der Konstruktoren und Destruktoren	39
3.38.3	Dokumentation der Elementfunktionen	40
3.39	MsgCard Klassenreferenz	40
3.39.1	Ausführliche Beschreibung	40
3.39.2	Beschreibung der Konstruktoren und Destruktoren	40
3.39.3	Dokumentation der Elementfunktionen	40

3.40	MsgCardRequest Klassenreferenz	40
3.40.1	Ausführliche Beschreibung	41
3.41	MsgGameEnd Klassenreferenz	41
3.41.1	Ausführliche Beschreibung	41
3.42	MsgMultiCards Klassenreferenz	41
3.42.1	Ausführliche Beschreibung	41
3.42.2	Beschreibung der Konstruktoren und Destruktoren	41
3.42.3	Dokumentation der Elementfunktionen	41
3.43	MsgMultiCardsRequest Klassenreferenz	42
3.43.1	Dokumentation der Elementfunktionen	42
3.44	MsgMultipleCardsRequest Klassenreferenz	42
3.45	MsgNumber Klassenreferenz	42
3.45.1	Ausführliche Beschreibung	42
3.45.2	Beschreibung der Konstruktoren und Destruktoren	43
3.45.3	Dokumentation der Elementfunktionen	44
3.46	MsgNumberRequest Klassenreferenz	44
3.46.1	Beschreibung der Konstruktoren und Destruktoren	44
3.47	MsgSelection Klassenreferenz	44
3.47.1	Ausführliche Beschreibung	44
3.47.2	Beschreibung der Konstruktoren und Destruktoren	45
3.47.3	Dokumentation der Elementfunktionen	46
3.48	MsgSelectionRequest Klassenreferenz	46
3.48.1	Ausführliche Beschreibung	46
3.49	MsgUser Klassenreferenz	46
3.49.1	Ausführliche Beschreibung	46
3.49.2	Beschreibung der Konstruktoren und Destruktoren	46
3.49.3	Dokumentation der Elementfunktionen	47
3.50	RulesetMessage Klassenreferenz	47
3.50.1	Ausführliche Beschreibung	47
3.50.2	Dokumentation der Elementfunktionen	47
3.51	Card Schnittstellenreferenz	48
3.51.1	Dokumentation der Elementfunktionen	48
3.52	ClientHearts Klassenreferenz	48
3.52.1	Beschreibung der Konstruktoren und Destruktoren	48
3.52.2	Dokumentation der Elementfunktionen	49
3.53	ClientRuleset Klassenreferenz	49
3.53.1	Ausführliche Beschreibung	50
3.53.2	Beschreibung der Konstruktoren und Destruktoren	50
3.53.3	Dokumentation der Elementfunktionen	50
3.54	ClientWizard Klassenreferenz	53

3.54.1	Beschreibung der Konstruktoren und Destruktoren	54
3.54.2	Dokumentation der Elementfunktionen	54
3.55	Colour Enum-Referenz	55
3.56	GameClientUpdate Klassenreferenz	55
3.56.1	Ausführliche Beschreibung	55
3.56.2	Beschreibung der Konstruktoren und Destruktoren	56
3.56.3	Dokumentation der Elementfunktionen	57
3.57	GamePhase Enum-Referenz	58
3.58	GameState Klassenreferenz	58
3.58.1	Ausführliche Beschreibung	59
3.58.2	Beschreibung der Konstruktoren und Destruktoren	59
3.58.3	Dokumentation der Elementfunktionen	59
3.59	HeartsCard Enum-Referenz	62
3.59.1	Dokumentation der Elementfunktionen	62
3.60	HeartsData Klassenreferenz	63
3.61	OtherData Klassenreferenz	63
3.61.1	Dokumentation der Elementfunktionen	63
3.62	PlayerState Klassenreferenz	64
3.62.1	Ausführliche Beschreibung	65
3.62.2	Beschreibung der Konstruktoren und Destruktoren	65
3.62.3	Dokumentation der Elementfunktionen	65
3.63	RulesetType Enum-Referenz	66
3.64	ServerHearts Klassenreferenz	66
3.64.1	Ausführliche Beschreibung	66
3.64.2	Dokumentation der Elementfunktionen	66
3.65	ServerRuleset Klassenreferenz	67
3.65.1	Ausführliche Beschreibung	68
3.65.2	Beschreibung der Konstruktoren und Destruktoren	68
3.65.3	Dokumentation der Elementfunktionen	68
3.66	ServerWizard Klassenreferenz	73
3.66.1	Ausführliche Beschreibung	73
3.66.2	Dokumentation der Elementfunktionen	73
3.67	WizardCard Enum-Referenz	74
3.67.1	Dokumentation der Elementfunktionen	75
3.68	WizData Klassenreferenz	75
3.68.1	Dokumentation der Elementfunktionen	75
3.69	GameServer Klassenreferenz	77
3.69.1	Ausführliche Beschreibung	77
3.69.2	Beschreibung der Konstruktoren und Destruktoren	77
3.69.3	Dokumentation der Elementfunktionen	78

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

<b>Index</b>	<b>88</b>
--------------	-----------

## 1 Hierarchie-Verzeichnis

### 1.1 Klassenhierarchie

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

<b>ClientMain</b>	<b>8</b>
<b>ClientModel</b>	<b>8</b>
<b>ClientState</b>	<b>12</b>
<b>Card</b>	<b>13</b>
<b>ChooseCards</b>	<b>14</b>
<b>ChooseItem</b>	<b>14</b>
<b>CreateGame</b>	<b>14</b>
<b>DiscardPile</b>	<b>16</b>
<b>DrawDeck</b>	<b>16</b>
<b>Game</b>	<b>16</b>
<b>GameLobby</b>	<b>17</b>

GamePanel	19
InputNumber	20
Language	21
Lobby	21
Login	23
OtherPlayer	24
OwnHand	24
Password	25
ScoreWindow	26
ViewCard	26
Warning	27
ComBeenKicked	29
ComChatMessage	29
ComClientLeave	30
ComClientQuit	30
ComCreateGameRequest	30
ComInitGameLobby	32
ComInitLobby	33
ComJoinRequest	33
ComKickPlayerRequest	35
ComLobbyUpdateGamelist	35
ComLoginRequest	36
ComRuleset	37
ComServerAcknowledgement	38
ComStartGame	38
ComUpdatePlayerlist	38
ComWarning	39
MsgCardRequest	40
RulesetMessage	47
MsgCard	40
MsgGameEnd	41
MsgMultiCards	41



<b>MsgMultiCardsRequest</b>	<b>42</b>
<b>MsgMultipleCardsRequest</b>	<b>42</b>
<b>MsgNumber</b>	<b>42</b>
<b>MsgNumberRequest</b>	<b>44</b>
<b>MsgSelection</b>	<b>44</b>
<b>MsgSelectionRequest</b>	<b>46</b>
<b>MsgUser</b>	<b>46</b>
<b>Card</b>	<b>48</b>
<b>HeartsCard</b>	<b>62</b>
<b>WizardCard</b>	<b>74</b>
<b>ClientRuleset</b>	<b>49</b>
<b>ClientHearts</b>	<b>48</b>
<b>ClientWizard</b>	<b>53</b>
<b>Colour</b>	<b>55</b>
<b>GameClientUpdate</b>	<b>55</b>
<b>GamePhase</b>	<b>58</b>
<b>GameState</b>	<b>58</b>
<b>OtherData</b>	<b>63</b>
<b>HeartsData</b>	<b>63</b>
<b>WizData</b>	<b>75</b>
<b>PlayerState</b>	<b>64</b>
<b>RulesetType</b>	<b>66</b>
<b>ServerRuleset</b>	<b>67</b>
<b>ServerHearts</b>	<b>66</b>
<b>ServerWizard</b>	<b>73</b>
<b>GameServerRepresentation</b>	<b>80</b>
<b>LobbyServer.ClientListenerThread</b>	<b>83</b>
<b>Player</b>	<b>84</b>
<b>Server</b>	<b>85</b>
<b>GameServer</b>	<b>77</b>
<b>LobbyServer</b>	<b>81</b>
<b>ServerMain</b>	<b>87</b>

## 2 Klassen-Verzeichnis

### 2.1 Auflistung der Klassen

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

<b>ClientMain</b>	Die <b>ClientMain</b> Klasse startet den Spielclient und initialisiert dessen Komponenten	8
<b>ClientModel</b>	Implementiert das Client Model	8
<b>ClientState</b>	Dieser Enumerator enthält alle Zustände in denen sich der Client befinden kann	12
<b>Card</b>	<b>Card</b> ist die View-seitige Repräsentation einer Karte	13
<b>ChooseCards</b>	In diesem Fenster muss der Benutzer eine vorbestimmte Menge Karten auswählen	14
<b>ChooseItem</b>	Dieses Fenster ermöglicht es dem Spieler aus einer Liste von Items eines auszuwählen	14
<b>CreateGame</b>	Das Fenster <b>CreateGame</b> dient dem Benutzer zur Erstellung eines neuen Spieles	14
<b>DiscardPile</b>	Stellt einen Ablagestapel dar, dieser kann sowohl für jeden Spieler einzeln oder für alle Spieler gemeinsam in der Mitte des Spielfeldes angezeigt werden	16
<b>DrawDeck</b>	Stellt einen Aufnahmestapel dar	16
<b>Game</b>	Im <b>Game</b> Fenster läuft das Spiel ab. Es enthält den Spielchat und ein <b>GamePanel</b>	16
<b>GameLobby</b>	Die <b>GameLobby</b> modelliert das Wartefenster, in dem beigetretene Spieler auf den Start des Spieles durch den Spielleiter warten	17
<b>GamePanel</b>	Das Panel ist die Komponente des Game-Fensters, welche das eigentliche Spiel darstellt	19
<b>InputNumber</b>	In diesem Fenster, kann der Benutzer eine Zahl eingeben	20
<b>Language</b>	<b>Language</b> stellt Repräsentationen verschiedener Sprachen dar, die von der GUI verwendet werden, um festzustellen welche Anzeigesprache verwendet werden soll	21
<b>Lobby</b>	Diese Klasse erzeugt die Ansicht der ServerLobby auf der Client Seite, in der die Spieler neue Spiele erstellen oder offenen beitreten können	21
<b>Login</b>	Das Login-Fenster repräsentiert den initialen Dialog zwischen Benutzer und Client	23

<b>OtherPlayer</b>	
Zeigt die Informationen über die anderen Spieler an, also den Namen, ein Symbol für die verdeckte Hand und das Label für zusätzliche Angaben	24
<b>OwnHand</b>	
Stellt die Karten dar, die der Spieler auf der Hand hat	24
<b>Password</b>	
Dieses Fenster ermöglicht die Eingabe eines Passwortes um einem Passwortgeschütztem Spiel beizutreten oder per 'Leave' wieder in die Lobby zurückzukehren	25
<b>ScoreWindow</b>	
Dieses Fenster zeigt den momentanen Punktestand nach jeder Runde und den Gesamtpunktestand am Ende des Spieles an	26
<b>ViewCard</b>	
Card ist die View-seitige Repräsentation einer Karte	26
<b>Warning</b>	
Das Warning-Fenster zeigt dem Benutzer Fehlermeldungen bzw	27
<b>ComBeenKicked</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	29
<b>ComChatMessage</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	29
<b>ComClientLeave</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	30
<b>ComClientQuit</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	30
<b>ComCreateGameRequest</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	30
<b>ComInitGameLobby</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	32
<b>ComInitLobby</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	33
<b>ComJoinRequest</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	33
<b>ComKickPlayerRequest</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	35
<b>ComLobbyUpdateGamelist</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	35
<b>ComLoginRequest</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	36
<b>ComRuleset</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	37
<b>ComServerAcknowledgement</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	38

<b>ComStartGame</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	38
<b>ComUpdatePlayerlist</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	38
<b>ComWarning</b>	
Diese Klasse ist ein spezielles Kommunikations-Objekt	39
<b>MsgCard</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	40
<b>MsgCardRequest</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	40
<b>MsgGameEnd</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	41
<b>MsgMultiCards</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	41
<b>MsgMultiCardsRequest</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	42
<b>MsgMultipleCardsRequest</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	42
<b>MsgNumber</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	42
<b>MsgNumberRequest</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	44
<b>MsgSelection</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	44
<b>MsgSelectionRequest</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	46
<b>MsgUser</b>	
Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse	46
<b>RulesetMessage</b>	
Diese Klasse ist eine Verfeinerung der ComRuleset-Klasse	47
<b>Card</b>	
Dieses Interface modelliert eine Spielkarte	48
<b>ClientHearts</b>	
Diese Klasse bildet das Regelwerk für den Clientmodel bei einer Partie Hearts	48
<b>ClientRuleset</b>	
<b>ClientRuleset</b> ist eine abstrakte Klasse und wird zur Regelvorauswertung im Client verwendet	49
<b>ClientWizard</b>	
Diese Klasse bildet das Regelwerk für den Client bei einer Partie Wizard	53
<b>Colour</b>	
Repräsentiert die Farbe einer Karte	55

<b>GameClientUpdate</b>	
Das <b>GameClientUpdate</b> wird vom <b>RuleSet</b> über den <b>GameServer</b> an den Client geschickt und enthält alle Änderungen des <b>GameState</b> , die für den Client relevant sind	55
<b>GamePhase</b>	
Die <b>GamePhase</b> modelliert die verschiedenen Zustände des Spiels im <b>GameState</b>	58
<b>GameState</b>	
Das <b>GameState</b> modelliert einen aktuellen Spielzustand, es wird vom <b>GameServer</b> instanziiert und vom <b>RuleSet</b> bearbeitet	58
<b>HeartsCard</b>	
Modelliert eine Heartskarte,	62
<b>HeartsData</b>	
Die Otherdata eines Spielers zum Spiel Hearts	63
<b>OtherData</b>	
<b>OtherData</b> speichert alle Spielinformationen eines Spielers, außer seiner eigenen Spielhand	63
<b>PlayerState</b>	
Repräsentiert den Spielzustand eines Spielers, und wird unter anderem im <b>GameState</b> gespeichert	64
<b>RulesetType</b>	
Die verschiedenen Regelwerke	66
<b>ServerHearts</b>	
Diese Klasse erstellt das Regelwerk zum Spiel Hearts	66
<b>ServerRuleset</b>	
Das <b>ServerRuleset</b> ist eine abstrakte Klasse und für den Ablauf und die Einhaltung der Regeln eines Spiels zuständig (/L280/)	67
<b>ServerWizard</b>	
Diese Klasse erstellt das Regelwerk zum Spiel Wizard	73
<b>WizardCard</b>	
Modelliert eine Wizardkarte	74
<b>WizData</b>	
Die Otherdata eines Spielers zum Spiel Wizard	75
<b>GameServer</b>	
Diese Klasse ist für die Verwaltung eines Spieles zuständig	77
<b>GameServerRepresentation</b>	
Dies eine Klasse, die Informationen über den Zustand eines Spielers bereithält	80
<b>LobbyServer</b>	
Diese Klasse ist für die Verwaltung der Spiellobby auf dem <b>Server</b> verantwortlich	81
<b>LobbyServer.ClientListenerThread</b>	
Diese innere Klasse ist für das Zustandekommen von Clientverbindungen zuständig	83
<b>Player</b>	
Die <b>Player</b> -Klasse wird zum Versenden von Java Serializable Objects, sowie zum Annehmen solcher verwendet	84
<b>Server</b>	
Ist ein abstrakte Klasse, von der die Klassen <b>LobbyServer</b> und <b>GameServer</b> 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

- void [clientQuit](#) ()
- void [leaveGameLobby](#) ()
- 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](#) ()
- Set< [GameServerRepresentation](#) > [getLobbyGamelist](#) ()
- final List< [Card](#) > [discardPile](#) ()
- final List< [Card](#) > [getOwnHand](#) ()
- final List< [OtherData](#) > [getOtherPlayerData](#) ()
- int [getOwnScore](#) ()
- void [setLanguage](#) (final [Language](#) language)
- [Language](#) [getLanguage](#) ()
- void [kickPlayer](#) (final String name)
- void [hostGame](#) (String gameName, String password, [RulesetType](#) type, Boolean hasPassword)
- void [sendMessage](#) (ComObject object)
- void [send](#) ([RulesetMessage](#) msg)

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

#### Private Methoden

- 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 bedient den Server durch den ListenerThread und leitet Daten an das Regelwerk und View weiter.

### 3.2.2 Dokumentation der Elementfunktionen

#### 3.2.2.1 void leaveGameLobby ( )

Schickt eine Nachricht an den Server die ihn darüber informiert, dass der Spieler die Spiellobby verlassen hat. Informiert ebenfalls die Observer über das Verlassen des Spieles.

#### 3.2.2.2 void receiveMessage ( `ComServerAcknowledgement` *ack* )

Diese Hilfsmethode 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.
------------	------------------------------------

### 3.2.2.3 List<String> getPlayerlist ( )

Diese Methode wird von der View beim betreten der Spiellobby aufgerufen und liefert eine Liste von Spielern in der Spiellobby.

#### Rückgabe

List Eine Liste der Spieler in der Spiellobby.

### 3.2.2.4 Set<GameServerRepresentation> getLobbyGamelist ( )

Diese Methode wird von der View beim betreten der Serverlobby aufgerufen und liefert eine Liste von Spielern und Spielen in der Serverlobby.

#### Rückgabe

Set Enthält alle Spiele in der ServerLobby.

### 3.2.2.5 final List<Card> discardPile ( )

Gibt der View die gespielte Karte eines anderen Spielers zurück.

#### Rückgabe

enum CardID. Die Id der Karte

### 3.2.2.6 final List<Card> getOwnHand ( )

Gibt der View die eigenen Spielkarten zurück.

#### Parameter

Card[]	Ein Array mit allen Karten, die man auf der Hand hat.
--------	---

### 3.2.2.7 final List<OtherData> getOtherPlayerData ( )

Liefert die otherData attribute anderer Spieler zurück.

#### Rückgabe

List<OtherData> Liste mit gespielten Karten.

### 3.2.2.8 int getOwnScore ( )

Gibt den Punktestand des Spielers aus.

#### Rückgabe

int Der eigene Punktestand.

### 3.2.2.9 void setLanguage ( final Language language )

Setzt die Sprache der GUI.



**Parameter**

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

Benutzt ClientModel.language.

**3.2.2.10 Language getLanguage ( )**

Liefert die Sprache der GUI.

**Rückgabe**

language Enumerator der die Spielsprache anzeigt.

Benutzt ClientModel.language.

**3.2.2.11 void kickPlayer ( final String name )**

Wird vom Controller aufgerufen um einen Spieler aus der Spiellobby zu entfernen.

**Parameter**

<i>name</i>	des Spielers welcher entfernt werden soll.
-------------	--

**3.2.2.12 void hostGame ( String gameName, String password, RulesetType type, Boolean hasPassword )**

Wird vom ClientController aufgerufen und erstellt ein neues Spiel auf dem Server.

**Parameter**

<i>gameName</i>	String Name des Spieles.
<i>password</i>	String Passwort zum sichern des Spieles.
<i>type</i>	RulesetType
<i>hasPassword</i>	

**3.2.2.13 int getPlayerCount ( )**

Die die Anzahl der Spieler eines Spieles zurück.

**Rückgabe**

int Die Spielerzahl eines Spieles.

**3.2.2.14 String getWindowText ( )**

Gibt.

**Rückgabe**

String

**3.2.2.15 void joinGame ( GameServerRepresentation game, String password )**

Diese Methode wird von dem ClientController aufgerufen um einem bereits erstelltem Spiel beizutreten.

**Parameter**

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

### 3.2.2.16 void initGame ( ) [private]

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

Es werden die Observer benachrichtigt und die View wechselt in die Spielansicht.

### 3.2.2.17 void makeMove ( Card card )

Wird vom ClientConroller aufgerufen um eine Karte auszuspielen.

Parameter

<i>id</i>	Die id der gespielten Karte um sie einer logischen Karte zuordnen zu können.
-----------	--

### 3.2.2.18 void informView ( ViewNotification note ) [private]

Hilfsmethode die alle verbundenen Observer der GUI kontaktiert.

Parameter

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

### 3.2.2.19 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.

### 3.2.2.20 String getWarningText ( )

Gibt den Text aus der bei einer Spielwarnung angezeigt wird.

Rückgabe

String Text der Warnung.

### 3.2.2.21 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

s	Pfadangabe zum zu zeichnenden Bild
n	ID der Karte

#### 3.4.3 Dokumentation der Elementfunktionen

##### 3.4.3.1 int getID ( )

Gibt die ID der Karte zurück.

#### 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 <a href="#">ClientModel</a>
<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 arg übergebenen Befehl wird ein Update des Fensters ausgeführt oder eine Fehlermeldung angezeigt.

#### Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<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 <code>MouseListener</code>
----------	--------------------------------

**3.7.2.2 void addRulesetSelectionListener ( ItemListener i )**

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

**Parameter**

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

**3.7.2.3 void addCreateButtonListener ( ActionListener a )**

Fügt einen ActionListener für den 'Create' Button hinzu.

**Parameter**

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

**3.7.2.4 void addLeaveButtonListener ( ActionListener a )**

Fügt einen ActionListener für den 'Leave' Button hinzu.

**Parameter**

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

**3.7.2.5 void setLanguage ( Language l )**

Ändert die Sprache des Fensters.

**Parameter**

<i>l</i>	Sprache in Form des <code>Language</code> -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 update (Observable o, Object arg)`
- `void update (Observable o, String arg)`

**Öffentliche, statische Methoden**

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

## 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 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 <a href="#">ClientModel</a>
<i>arg</i>	erwartet: playedCardsUpdate, otherDataUpdate, moveAcknowledged, gameStarted

## 3.10.3.2 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 <a href="#">ClientModel</a>
<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 <a href="#">ClientModel</a>
<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() übergeben wird.

Dieser wird als Chatnachricht interpretiert und dem Chatlog angefügt.

## Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<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 <a href="#">ClientModel</a>
----------	--

<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

a	ein ActionListener
---	--------------------

#### 3.15.2.2 void addHostButtonListener ( ActionListener a )

Fügt einen ActionListener für den 'Host' Button hinzu.

Parameter

a	ein ActionListener
---	--------------------

#### 3.15.2.3 void addLeaveButtonListener ( ActionListener a )

Fügt einen ActionListener für den 'Leave' Button hinzu.

Parameter

a	ein ActionListener
---	--------------------

#### 3.15.2.4 void addChatMessageListener ( KeyListener k )

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

Parameter

k	
---	--

#### 3.15.2.5 void setLanguage ( Language l )

Ändert die Sprache des Fensters.

Parameter

l	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 <a href="#">ClientModel</a>
<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() Ã¼bergeben wird.

Dieser wird als Chatnachricht interpretiert und dem Chatlog angefügt.

##### Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<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 <a href="#">ClientModel</a>
<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**

/	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 <a href="#">ClientModel</a>
<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 <a href="#">ClientModel</a>
<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

s	Pfadangabe zum zu zeichnenden Bild
n	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 <a href="#">ClientModel</a>
<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 [getName](#) ()
- 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.ComCreateGameRequest()`.

### 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 Konstruktor 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 Kontruktor für eine neue ComKickPlayerRequest-Nachricht.

Diese enthält den Namen des Spielers, der aus den 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 gelöscht 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 Regelwerkaufwurfes 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&lt;Card&gt; 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>gameClientUpdate</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&lt; Card &gt; 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 <code>ClientModel</code> 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**

Der	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&lt;Card&gt; getOwnHand ( ) [protected]

Holt die Karten die der Client auf der Hand hat.

## Rückgabe

ownHand Die Hand des Clients

## 3.56.3.2 Map&lt;String, Card&gt; 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&lt;Card&gt; 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 GameServerinstanz 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**

Der	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 <a href="#">LobbyServer</a> der den <a href="#">GameServer</a> 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 <a href="#">Player</a> , 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 <a href="#">Player</a> , 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 <code>ComObject</code> , 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 <code>ComObject</code> , 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 <a href="#">GameServer</a> der eingefügt wird
-------------	---

**3.71.2.4 void removeGameServer ( GameServer game )**

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

**Parameter**

<i>game</i>	ist der <a href="#">GameServer</a> 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 change-Server Methode hinzu.

Der neue [GameServer](#) wird in das gameServerSet eingefügt. Durch broadcast wird im [LobbyServer](#) sowohl Com-UpdatePlayerlist 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 <a href="#">Player</a> 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 <a href="#">Player</a> 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 <a href="#">LobbyServer</a> , der zu Beginn den <a href="#">Player</a> 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 ComObjekt 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 <a href="#">Server</a>
------------------	-------------------------------------

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 <a href="#">Player</a> von dem die Nachricht kommt
<i>com</i>	ist das ComObjekt 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 <a href="#">Player</a> 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 <a href="#">Player</a> , 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 <a href="#">Player</a> , 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, 19
  - Client::View::Lobby, 22
- addConnectButtonListener
  - Client::View::Login, 24
- addCreateButtonListener
  - Client::View::CreateGame, 16
- addGameServer
  - Server::LobbyServer, 82
- addHostButtonListener
  - Client::View::Lobby, 22
- addJoinButtonListener
  - Client::View::Lobby, 22
  - Client::View::Password, 25
- addLanguageSelectionListener
  - Client::View::Login, 24
- addLeaveButtonListener
  - Client::View::CreateGame, 16
  - Client::View::GameLobby, 18
  - Client::View::Lobby, 22
- addName
  - Server::LobbyServer, 81
- addPanelMouseListener
  - Client::View::CreateGame, 15
- addPlayer
  - Server::GameServer, 78
  - Server::Server, 86
- addPlayerToGame
  - Ruleset::GameState, 59
  - Ruleset::ServerRuleset, 70
- addRemoveButtonListener
  - Client::View::GameLobby, 18
- addRulesetSelectionListener
  - Client::View::CreateGame, 16
- addStartButtonListener
  - Client::View::GameLobby, 18
- broadcast
  - Ruleset::ServerRuleset, 72
  - Server::Server, 86
- Card, 13, 48
  - Client::View::Card, 13
- changeServer
  - Server::Player, 85
- ChooseCards, 14
- ChooseItem, 14
- Client::ClientMain
  - main, 8
- Client::ClientModel
  - createConnection, 12
  - discardPile, 10
  - getLanguage, 11
  - getLobbyGamelist, 10
  - getOtherPlayerData, 10
  - getOwnHand, 10
  - getOwnScore, 10
  - getPlayerCount, 11
  - getPlayerlist, 9
  - getRulesets, 12
  - getWarningText, 12
  - getWindowText, 11
  - hostGame, 11
  - informView, 12
  - initGame, 12
  - joinGame, 11
  - kickPlayer, 11
  - leaveGameLobby, 9
  - makeMove, 12
  - receiveMessage, 9
  - setLanguage, 10
- Client::View::Card
  - Card, 13
  - getID, 13
- Client::View::ChooseCards
  - update, 14
- Client::View::ChooseItem
  - update, 14
- Client::View::CreateGame
  - addCreateButtonListener, 16
  - addLeaveButtonListener, 16
  - addPanelMouseListener, 15
  - addRulesetSelectionListener, 16
  - setLanguage, 16
- Client::View::Game
  - Game, 17
  - update, 17
- Client::View::GameLobby
  - addChatMessageListener, 19
  - addLeaveButtonListener, 18
  - addRemoveButtonListener, 18
  - addStartButtonListener, 18
  - setLanguage, 19
  - update, 19
- Client::View::GamePanel
  - setupTrickGame, 20
- Client::View::InputNumber
  - update, 20
- Client::View::Lobby
  - addChatMessageListener, 22
  - addHostButtonListener, 22
  - addJoinButtonListener, 22
  - addLeaveButtonListener, 22
  - setLanguage, 22
  - update, 22, 23
- Client::View::Login
  - addConnectButtonListener, 24
  - addLanguageSelectionListener, 24
  - setLanguage, 24

- update, [24](#)
- Client::View::Password
  - addJoinButtonListener, [25](#)
  - setLanguage, [25](#)
  - update, [26](#)
- Client::View::ScoreWindow
  - update, [26](#)
- Client::View::ViewCard
  - getID, [27](#)
  - ViewCard, [27](#)
- Client::View::Warning
  - update, [27](#)
- ClientHearts, [48](#)
  - Ruleset::ClientHearts, [48](#)
- ClientMain, [8](#)
- ClientModel, [8](#)
- ClientRuleset, [49](#)
  - Ruleset::ClientRuleset, [50](#)
- ClientState, [12](#)
- ClientWizard, [53](#)
  - Ruleset::ClientWizard, [54](#)
- Colour, [55](#)
- ComBeenKicked, [29](#)
  - ComObjects::ComBeenKicked, [29](#)
- ComChatMessage, [29](#)
  - ComObjects::ComChatMessage, [30](#)
- ComClientLeave, [30](#)
- ComClientQuit, [30](#)
- ComCreateGameRequest, [30](#)
  - ComObjects::ComCreateGameRequest, [31](#)
- ComInitGameLobby, [32](#)
  - ComObjects::ComInitGameLobby, [32](#)
- ComInitLobby, [33](#)
  - ComObjects::ComInitLobby, [33](#)
- ComJoinRequest, [33](#)
  - ComObjects::ComJoinRequest, [34](#)
- ComKickPlayerRequest, [35](#)
  - ComObjects::ComKickPlayerRequest, [35](#)
- ComLobbyUpdateGamelist, [35](#)
  - ComObjects::ComLobbyUpdateGamelist, [36](#)
- ComLoginRequest, [36](#)
  - ComObjects::ComLoginRequest, [37](#)
- ComObjects::ComBeenKicked
  - ComBeenKicked, [29](#)
  - getMessage, [29](#)
- ComObjects::ComChatMessage
  - ComChatMessage, [30](#)
  - getChatMessage, [30](#)
- ComObjects::ComCreateGameRequest
  - ComCreateGameRequest, [31](#)
  - getGameName, [31](#)
  - getPassword, [32](#)
  - getRuleset, [31](#)
  - hasPassword, [31](#)
- ComObjects::ComInitGameLobby
  - ComInitGameLobby, [32](#)
  - getPlayerList, [32](#)
- ComObjects::ComInitLobby
  - ComInitLobby, [33](#)
  - getGameList, [33](#)
  - getPlayerList, [33](#)
- ComObjects::ComJoinRequest
  - ComJoinRequest, [34](#)
  - gameMasterName, [34](#)
  - getGameMasterName, [34](#)
- ComObjects::ComKickPlayerRequest
  - ComKickPlayerRequest, [35](#)
  - getPlayerName, [35](#)
- ComObjects::ComLobbyUpdateGamelist
  - ComLobbyUpdateGamelist, [36](#)
  - getGameServer, [36](#)
  - isRemoveFlag, [36](#)
- ComObjects::ComLoginRequest
  - ComLoginRequest, [37](#)
  - getPlayerName, [37](#)
- ComObjects::ComRuleset
  - ComRuleset, [37](#)
  - getRulesetMessage, [37](#)
- ComObjects::ComUpdatePlayerlist
  - ComUpdatePlayerlist, [38](#)
  - getPlayerName, [39](#)
  - isRemoveFlag, [39](#)
- ComObjects::ComWarning
  - ComWarning, [39](#)
  - getWarning, [40](#)
- ComObjects::MsgCard
  - getCard, [40](#)
  - MsgCard, [40](#)
- ComObjects::MsgMultiCards
  - getCardList, [41](#)
  - MsgMultiCards, [41](#)
- ComObjects::MsgMultiCardsRequest
  - getCount, [42](#)
- ComObjects::MsgNumber
  - getNumber, [44](#)
  - MsgNumber, [43](#)
- ComObjects::MsgNumberRequest
  - MsgNumberRequest, [44](#)
- ComObjects::MsgSelection
  - getSelection, [46](#)
  - MsgSelection, [45](#)
- ComObjects::MsgUser
  - getGameClientUpdate, [47](#)
  - MsgUser, [46](#)
- ComObjects::RulesetMessage
  - visit, [47](#)
- ComRuleset, [37](#)
  - ComObjects::ComRuleset, [37](#)
- ComServerAcknowledgement, [38](#)
- ComStartGame, [38](#)
- ComUpdatePlayerlist, [38](#)
  - ComObjects::ComUpdatePlayerlist, [38](#)
- ComWarning, [39](#)
  - ComObjects::ComWarning, [39](#)
- createConnection
  - Client::ClientModel, [12](#)

createDeck  
     Ruleset::ServerRuleset, 69  
 CreateGame, 14  
  
 dealCards  
     Ruleset::GameState, 61  
     Ruleset::ServerRuleset, 72  
 DiscardPile, 16  
 discardPile  
     Client::ClientModel, 10  
 DrawDeck, 16  
  
 Game, 16  
     Client::View::Game, 17  
 GameClientUpdate, 55  
     Ruleset::GameClientUpdate, 56  
 GameLobby, 17  
 gameMasterName  
     ComObjects::ComJoinRequest, 34  
 GamePanel, 19  
 GamePhase, 58  
 GameServer, 77  
     Server::GameServer, 77  
 GameServerRepresentation, 80  
     Server::GameServerRepresentation, 80  
 GameState, 58  
     Ruleset::GameState, 59  
 getAnnouncedTricks  
     Ruleset::WizData, 75  
 getCard  
     ComObjects::MsgCard, 40  
 getCardList  
     ComObjects::MsgMultiCards, 41  
 getCardsLeftInDeck  
     Ruleset::GameState, 60  
 getChatMessage  
     ComObjects::ComChatMessage, 30  
 getColour  
     Ruleset::Card, 48  
     Ruleset::HeartsCard, 62  
     Ruleset::WizardCard, 75  
 getCount  
     ComObjects::MsgMultiCardsRequest, 42  
 getCurrentPlayer  
     Ruleset::ClientRuleset, 51  
     Ruleset::GameClientUpdate, 57  
     Ruleset::GameState, 60  
     Ruleset::ServerRuleset, 70  
 getFirstPlayer  
     Ruleset::GameState, 59  
     Ruleset::ServerRuleset, 69  
 getGameClientUpdate  
     ComObjects::MsgUser, 47  
 getGameList  
     ComObjects::ComInitLobby, 33  
 getGameMasterName  
     ComObjects::ComJoinRequest, 34  
 getGameName  
     ComObjects::ComCreateGameRequest, 31  
  
 getGamePhase  
     Ruleset::ClientRuleset, 51  
     Ruleset::ServerRuleset, 69  
 getGameServer  
     ComObjects::ComLobbyUpdateGamelist, 36  
 getHand  
     Ruleset::PlayerState, 65  
 getID  
     Client::View::Card, 13  
     Client::View::ViewCard, 27  
 getLanguage  
     Client::ClientModel, 11  
 getLobbyGamelist  
     Client::ClientModel, 10  
 getMaxPlayers  
     Ruleset::ClientRuleset, 50  
     Ruleset::ServerRuleset, 68  
 getMessage  
     ComObjects::ComBeenKicked, 29  
 getMinPlayers  
     Ruleset::ClientRuleset, 50  
     Ruleset::ServerRuleset, 68  
 getName  
     Ruleset::PlayerState, 65  
     Server::Player, 85  
 getNumber  
     ComObjects::MsgNumber, 44  
 getNumberOfPlayedCards  
     Ruleset::GameState, 61  
 getNumberOfTricks  
     Ruleset::OtherData, 64  
 getOtherData  
     Ruleset::PlayerState, 65  
     Ruleset::ServerRuleset, 70  
 getOtherPlayerData  
     Client::ClientModel, 10  
     Ruleset::ClientRuleset, 51  
     Ruleset::GameClientUpdate, 57  
 getOwnData  
     Ruleset::ClientRuleset, 51  
     Ruleset::GameClientUpdate, 57  
 getOwnHand  
     Client::ClientModel, 10  
     Ruleset::ClientRuleset, 51  
     Ruleset::GameClientUpdate, 57  
 getOwnScore  
     Client::ClientModel, 10  
 getPassword  
     ComObjects::ComCreateGameRequest, 32  
 getPlayedCards  
     Ruleset::GameClientUpdate, 57  
     Ruleset::GameState, 60  
 getPlayerCards  
     Ruleset::GameState, 61  
     Ruleset::ServerRuleset, 70  
 getPlayerCount  
     Client::ClientModel, 11  
 getPlayerList



- ComObjects::ComInitGameLobby, 32
- ComObjects::ComInitLobby, 33
- getPlayerName
  - ComObjects::ComKickPlayerRequest, 35
  - ComObjects::ComLoginRequest, 37
  - ComObjects::ComUpdatePlayerlist, 39
- getPlayerState
  - Ruleset::GameState, 60
  - Ruleset::ServerRuleset, 70
- getPlayerlist
  - Client::ClientModel, 9
- getPoints
  - Ruleset::OtherData, 64
- getRoundNumber
  - Ruleset::GameState, 61
- getRuleset
  - ComObjects::ComCreateGameRequest, 31
- getRulesetMessage
  - ComObjects::ComRuleset, 37
- getRulesetType
  - Ruleset::ClientRuleset, 50
  - Ruleset::ServerRuleset, 68
- getRulesets
  - Client::ClientModel, 12
- getSelection
  - ComObjects::MsgSelection, 46
- getTrumpCard
  - Ruleset::ClientRuleset, 51
  - Ruleset::GameClientUpdate, 57
  - Ruleset::GameState, 60
- getValue
  - Ruleset::Card, 48
  - Ruleset::HeartsCard, 62
  - Ruleset::WizardCard, 75
- getWarning
  - ComObjects::ComWarning, 40
- getWarningText
  - Client::ClientModel, 12
- getWindowText
  - Client::ClientModel, 11
- giveACard
  - Ruleset::GameState, 61
  - Ruleset::ServerRuleset, 72
- handleIOException
  - Server::GameServer, 79
  - Server::LobbyServer, 83
  - Server::Server, 87
- hasPassword
  - ComObjects::ComCreateGameRequest, 31
- HeartsCard, 62
- HeartsData, 63
- hostGame
  - Client::ClientModel, 11
- informView
  - Client::ClientModel, 12
- initGame
  - Client::ClientModel, 12
- initLobby
  - Server::GameServer, 79
  - Server::LobbyServer, 83
- InputNumber, 20
- isRemoveFlag
  - ComObjects::ComLobbyUpdateGamelist, 36
  - ComObjects::ComUpdatePlayerlist, 39
- isValidMove
  - Ruleset::ClientHearts, 49
  - Ruleset::ClientRuleset, 53
  - Ruleset::ClientWizard, 54
  - Ruleset::ServerHearts, 66
  - Ruleset::ServerRuleset, 73
  - Ruleset::ServerWizard, 73
- joinGame
  - Client::ClientModel, 11
- kickPlayer
  - Client::ClientModel, 11
- Language, 21
- leaveGameLobby
  - Client::ClientModel, 9
- Lobby, 21
- LobbyServer, 81
- LobbyServer.ClientListenerThread, 83
- Login, 23
- madeTrick
  - Ruleset::OtherData, 63
- main
  - Client::ClientMain, 8
  - Server::ServerMain, 87
- makeMove
  - Client::ClientModel, 12
- MsgCard, 40
  - ComObjects::MsgCard, 40
- MsgCardRequest, 40
- MsgGameEnd, 41
- MsgMultiCards, 41
  - ComObjects::MsgMultiCards, 41
- MsgMultiCardsRequest, 42
- MsgMultipleCardsRequest, 42
- MsgNumber, 42
  - ComObjects::MsgNumber, 43
- MsgNumberRequest, 44
  - ComObjects::MsgNumberRequest, 44
- MsgSelection, 44
  - ComObjects::MsgSelection, 45
- MsgSelectionRequest, 46
- MsgUser, 46
  - ComObjects::MsgUser, 46
- nextPlayer
  - Ruleset::ServerRuleset, 69
- OtherData, 63
- OtherPlayer, 24
- OwnHand, 24

Password, 25  
 password  
     Server::GameServer, 80  
 playCard  
     Ruleset::GameState, 62  
     Ruleset::ServerRuleset, 72  
 Player, 84  
     Server::Player, 84  
 PlayerState, 64  
     Ruleset::PlayerState, 65  
  
 receiveMessage  
     Client::ClientModel, 9  
     Server::GameServer, 78, 79  
     Server::LobbyServer, 82, 83  
     Server::Server, 86  
 removeCard  
     Ruleset::PlayerState, 66  
 removeGameServer  
     Server::LobbyServer, 82  
 removeName  
     Server::LobbyServer, 82  
 removePlayer  
     Server::GameServer, 78  
     Server::Server, 86  
 removeTricks  
     Ruleset::OtherData, 64  
 resolveMessage  
     Ruleset::ClientHearts, 49  
     Ruleset::ClientRuleset, 51, 53  
     Ruleset::ClientWizard, 54, 55  
     Ruleset::ServerHearts, 67  
     Ruleset::ServerRuleset, 72  
     Ruleset::ServerWizard, 74  
 Ruleset::Card  
     getColour, 48  
     getValue, 48  
 Ruleset::ClientHearts  
     ClientHearts, 48  
     isValidMove, 49  
     resolveMessage, 49  
     send, 49  
 Ruleset::ClientRuleset  
     ClientRuleset, 50  
     getCurrentPlayer, 51  
     getGamePhase, 51  
     getMaxPlayers, 50  
     getMinPlayers, 50  
     getOtherPlayerData, 51  
     getOwnData, 51  
     getOwnHand, 51  
     getRulesetType, 50  
     getTrumpCard, 51  
     isValidMove, 53  
     resolveMessage, 51, 53  
     send, 53  
 Ruleset::ClientWizard  
     ClientWizard, 54  
     isValidMove, 54  
     resolveMessage, 54, 55  
     send, 54  
 Ruleset::GameClientUpdate  
     GameClientUpdate, 56  
     getCurrentPlayer, 57  
     getOtherPlayerData, 57  
     getOwnData, 57  
     getOwnHand, 57  
     getPlayedCards, 57  
     getTrumpCard, 57  
 Ruleset::GameState  
     addPlayerToGame, 59  
     dealCards, 61  
     GameState, 59  
     getCardsLeftInDeck, 60  
     getCurrentPlayer, 60  
     getFirstPlayer, 59  
     getNumberOfPlayedCards, 61  
     getPlayedCards, 60  
     getPlayerCards, 61  
     getPlayerState, 60  
     getRoundNumber, 61  
     getTrumpCard, 60  
     giveACard, 61  
     playCard, 62  
     setCurrentPlayer, 59  
     setFirstPlayer, 59  
     setTrumpCard, 60  
 Ruleset::HeartsCard  
     getColour, 62  
     getValue, 62  
 Ruleset::OtherData  
     getNumberOfTricks, 64  
     getPoints, 64  
     madeTrick, 63  
     removeTricks, 64  
     setPoints, 64  
 Ruleset::PlayerState  
     addCard, 65  
     getHand, 65  
     getName, 65  
     getOtherData, 65  
     PlayerState, 65  
     removeCard, 66  
 Ruleset::ServerHearts  
     isValidMove, 66  
     resolveMessage, 67  
 Ruleset::ServerRuleset  
     addPlayerToGame, 70  
     broadcast, 72  
     createDeck, 69  
     dealCards, 72  
     getCurrentPlayer, 70  
     getFirstPlayer, 69  
     getGamePhase, 69  
     getMaxPlayers, 68  
     getMinPlayers, 68  
     getOtherData, 70

- getPlayerCards, 70
- getPlayerState, 70
- getRulesetType, 68
- giveACard, 72
- isValidMove, 73
- nextPlayer, 69
- playCard, 72
- resolveMessage, 72
- send, 70
- ServerRuleset, 68
- setCurrentPlayer, 69
- setFirstPlayer, 69
- setTrumpCard, 72
- Ruleset::ServerWizard
  - isValidMove, 73
  - resolveMessage, 74
- Ruleset::WizData
  - getAnnouncedTricks, 75
  - setAnnounceTricks, 75
- Ruleset::WizardCard
  - getColour, 75
  - getValue, 75
- RulesetMessage, 47
- RulesetType, 66
- run
  - Server::Player, 84
- ScoreWindow, 26
- send
  - Ruleset::ClientHearts, 49
  - Ruleset::ClientRuleset, 53
  - Ruleset::ClientWizard, 54
  - Ruleset::ServerRuleset, 70
  - Server::Player, 85
- sendToPlayer
  - Server::Server, 86
- Server, 85
- Server::GameServer
  - addPlayer, 78
  - GameServer, 77
  - handleIOException, 79
  - initLobby, 79
  - password, 80
  - receiveMessage, 78, 79
  - removePlayer, 78
- Server::GameServerRepresentation
  - GameServerRepresentation, 80
- Server::LobbyServer
  - addGameServer, 82
  - addName, 81
  - handleIOException, 83
  - initLobby, 83
  - receiveMessage, 82, 83
  - removeGameServer, 82
  - removeName, 82
- Server::Player
  - changeServer, 85
  - getName, 85
  - Player, 84
  - run, 84
  - send, 85
  - setName, 85
- Server::Server
  - addPlayer, 86
  - broadcast, 86
  - handleIOException, 87
  - receiveMessage, 86
  - removePlayer, 86
  - sendToPlayer, 86
- Server::ServerMain
  - main, 87
- ServerHearts, 66
- ServerMain, 87
- ServerRuleset, 67
  - Ruleset::ServerRuleset, 68
- ServerWizard, 73
- setAnnounceTricks
  - Ruleset::WizData, 75
- setCurrentPlayer
  - Ruleset::GameState, 59
  - Ruleset::ServerRuleset, 69
- setFirstPlayer
  - Ruleset::GameState, 59
  - Ruleset::ServerRuleset, 69
- setLanguage
  - Client::ClientModel, 10
  - Client::View::CreateGame, 16
  - Client::View::GameLobby, 19
  - Client::View::Lobby, 22
  - Client::View::Login, 24
  - Client::View::Password, 25
- setName
  - Server::Player, 85
- setPoints
  - Ruleset::OtherData, 64
- setTrumpCard
  - Ruleset::GameState, 60
  - Ruleset::ServerRuleset, 72
- setupTrickGame
  - Client::View::GamePanel, 20
- update
  - Client::View::ChooseCards, 14
  - Client::View::ChooseItem, 14
  - Client::View::Game, 17
  - Client::View::GameLobby, 19
  - Client::View::InputNumber, 20
  - Client::View::Lobby, 22, 23
  - Client::View::Login, 24
  - Client::View::Password, 26
  - Client::View::ScoreWindow, 26
  - Client::View::Warning, 27
- ViewCard, 26
  - Client::View::ViewCard, 27
- visit
  - ComObjects::RulesetMessage, 47

Warning, [27](#)

WizData, [75](#)

WizardCard, [74](#)