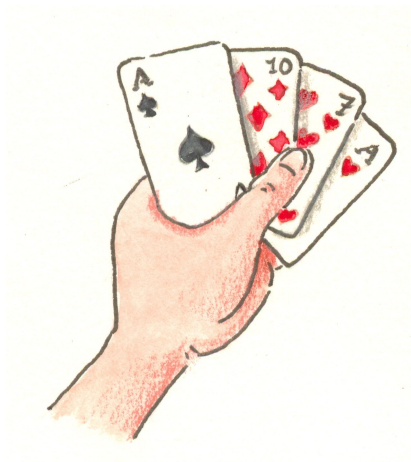


# SPEZIFIKATION

13. November 2013



## NET-WIZHEARTS

Phase	Verantwortlicher	E-Mail
Pflichtenheft	Alina Meixl	alina@meixl.de
Entwurf	Viktoria Witka	witkaviktoria@freenet.de
Spezifikation	Daniel Riedl	dariedl14@yahoo.de
Implementation	Andreas Altenbuchner	a.andi007@gmail.com
Verifikation	Patrick Kubin	kubin@fim.uni-passau.de
Präsentation	w	w

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Änderungen im Vergleich zum Entwurf</b>	<b>2</b>
2.1	Server	2
2.1.1	Server	2
2.2	Client	2
2.2.1	Language	2
2.2.2	MVMessage	2
2.2.3	ClientState-Enum	2
2.3	ClientView	2
2.3.1	MVMessages	2
2.3.2	ViewCard	2
2.4	Ruleset	2
2.4.1	ServerRuleset	2
2.4.2	ClientRuleset	3
2.4.3	Card	3
2.4.4	Colour	3
2.4.5	RulesetType	3
2.5	ComObjects	3
2.5.1	ComObject	3
2.5.2	RulesetMessage	3
2.5.3	ComLogin	3
2.5.4	ComKickPlayer	3
2.5.5	GameClientUpdate	3
2.5.6	commands	3
2.5.7	types	3
2.5.8	ComBeenKicked	4
2.5.9	ComClientLeave	4
2.5.10	ComClientQuit	4
2.5.11	ComCreateGameRequest	4
2.5.12	ComServerAcknowledgement	4
2.5.13	ComStartGame	4
2.5.14	ComWarning	4
2.5.15	MsgCardRequest	4
2.5.16	MsgMultiCardsRequest	4
2.5.17	MsgNumberRequest	4
2.5.18	MsgSelectionRequest	4
2.5.19	ComObjects-Klassen allgemein	5

<b>3</b>	<b>Hierarchie-Verzeichnis</b>	<b>5</b>
3.1	Klassenhierarchie . . . . .	5
<b>4</b>	<b>Klassen-Verzeichnis</b>	<b>7</b>
4.1	Auflistung der Klassen . . . . .	7
<b>5</b>	<b>Klassen-Dokumentation</b>	<b>11</b>
5.1	Client.ClientController Klassenreferenz . . . . .	11
5.1.1	Ausführliche Beschreibung . . . . .	11
5.2	Client.ClientMain Klassenreferenz . . . . .	12
5.2.1	Ausführliche Beschreibung . . . . .	12
5.2.2	Dokumentation der Elementfunktionen . . . . .	12
5.3	Client.ClientModel Klassenreferenz . . . . .	12
5.3.1	Ausführliche Beschreibung . . . . .	13
5.3.2	Beschreibung der Konstruktoren und Destruktoren . . . . .	13
5.3.3	Dokumentation der Elementfunktionen . . . . .	14
5.4	Client.ClientState Enum-Referenz . . . . .	20
5.4.1	Ausführliche Beschreibung . . . . .	21
5.5	Client.MessageListenerThread Klassenreferenz . . . . .	21
5.5.1	Ausführliche Beschreibung . . . . .	21
5.5.2	Dokumentation der Elementfunktionen . . . . .	21
5.6	Client.View.ChooseCards Klassenreferenz . . . . .	21
5.6.1	Ausführliche Beschreibung . . . . .	22
5.6.2	Dokumentation der Elementfunktionen . . . . .	22
5.7	Client.View.ChooseItem Klassenreferenz . . . . .	22
5.7.1	Ausführliche Beschreibung . . . . .	22
5.7.2	Dokumentation der Elementfunktionen . . . . .	22
5.8	Client.View.CreateGame Klassenreferenz . . . . .	23
5.8.1	Ausführliche Beschreibung . . . . .	23
5.8.2	Dokumentation der Elementfunktionen . . . . .	23
5.9	Client.View.DiscardPile Klassenreferenz . . . . .	24
5.9.1	Ausführliche Beschreibung . . . . .	24
5.10	Client.View.DrawDeck Klassenreferenz . . . . .	24
5.10.1	Ausführliche Beschreibung . . . . .	24
5.11	Client.View.Game Klassenreferenz . . . . .	24
5.11.1	Ausführliche Beschreibung . . . . .	25
5.11.2	Beschreibung der Konstruktoren und Destruktoren . . . . .	25
5.11.3	Dokumentation der Elementfunktionen . . . . .	25
5.12	Client.View.GameLobby Klassenreferenz . . . . .	26
5.12.1	Ausführliche Beschreibung . . . . .	27
5.12.2	Dokumentation der Elementfunktionen . . . . .	27

5.13	Client.View.GamePanel Klassenreferenz	28
5.13.1	Ausführliche Beschreibung	28
5.14	Client.View.InputNumber Klassenreferenz	28
5.14.1	Ausführliche Beschreibung	29
5.14.2	Dokumentation der Elementfunktionen	29
5.15	Client.View.Language Enum-Referenz	29
5.15.1	Ausführliche Beschreibung	29
5.16	Client.View.Lobby Klassenreferenz	29
5.16.1	Ausführliche Beschreibung	30
5.16.2	Dokumentation der Elementfunktionen	30
5.17	Client.View.Login Klassenreferenz	31
5.17.1	Ausführliche Beschreibung	32
5.17.2	Dokumentation der Elementfunktionen	32
5.18	Client.View.OtherPlayer Klassenreferenz	33
5.18.1	Ausführliche Beschreibung	33
5.19	Client.View.OwnHand Klassenreferenz	33
5.19.1	Ausführliche Beschreibung	33
5.20	Client.View.Password Klassenreferenz	33
5.20.1	Ausführliche Beschreibung	34
5.20.2	Dokumentation der Elementfunktionen	34
5.21	Client.View.ScoreWindow Klassenreferenz	34
5.21.1	Ausführliche Beschreibung	35
5.21.2	Dokumentation der Elementfunktionen	35
5.22	Client.View.ViewCard Klassenreferenz	35
5.22.1	Ausführliche Beschreibung	35
5.22.2	Beschreibung der Konstruktoren und Destruktoren	35
5.22.3	Dokumentation der Elementfunktionen	36
5.23	Client.View.Warning Klassenreferenz	36
5.23.1	Ausführliche Beschreibung	36
5.23.2	Dokumentation der Elementfunktionen	36
5.24	Client.View.Notification Enum-Referenz	37
5.24.1	Ausführliche Beschreibung	37
5.25	ComObjects.ComBeenKicked Klassenreferenz	37
5.25.1	Ausführliche Beschreibung	37
5.25.2	Beschreibung der Konstruktoren und Destruktoren	38
5.25.3	Dokumentation der Elementfunktionen	38
5.26	ComObjects.ComChatMessage Klassenreferenz	38
5.26.1	Ausführliche Beschreibung	39
5.26.2	Beschreibung der Konstruktoren und Destruktoren	39
5.26.3	Dokumentation der Elementfunktionen	39

5.27 ComObjects.ComClientLeave Klassenreferenz . . . . .	39
5.27.1 Ausführliche Beschreibung . . . . .	40
5.27.2 Dokumentation der Elementfunktionen . . . . .	40
5.28 ComObjects.ComClientQuit Klassenreferenz . . . . .	40
5.28.1 Ausführliche Beschreibung . . . . .	40
5.28.2 Dokumentation der Elementfunktionen . . . . .	40
5.29 ComObjects.ComCreateGameRequest Klassenreferenz . . . . .	41
5.29.1 Ausführliche Beschreibung . . . . .	41
5.29.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	41
5.29.3 Dokumentation der Elementfunktionen . . . . .	42
5.30 ComObjects.ComInitGameLobby Klassenreferenz . . . . .	43
5.30.1 Ausführliche Beschreibung . . . . .	43
5.30.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	43
5.30.3 Dokumentation der Elementfunktionen . . . . .	43
5.31 ComObjects.ComInitLobby Klassenreferenz . . . . .	44
5.31.1 Ausführliche Beschreibung . . . . .	44
5.31.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	44
5.31.3 Dokumentation der Elementfunktionen . . . . .	45
5.32 ComObjects.ComJoinRequest Klassenreferenz . . . . .	45
5.32.1 Ausführliche Beschreibung . . . . .	46
5.32.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	46
5.32.3 Dokumentation der Elementfunktionen . . . . .	46
5.32.4 Dokumentation der Datenelemente . . . . .	46
5.33 ComObjects.ComKickPlayerRequest Klassenreferenz . . . . .	47
5.33.1 Ausführliche Beschreibung . . . . .	47
5.33.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	47
5.33.3 Dokumentation der Elementfunktionen . . . . .	47
5.34 ComObjects.ComLobbyUpdateGamelist Klassenreferenz . . . . .	49
5.34.1 Ausführliche Beschreibung . . . . .	49
5.34.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	49
5.34.3 Dokumentation der Elementfunktionen . . . . .	50
5.35 ComObjects.ComLoginRequest Klassenreferenz . . . . .	50
5.35.1 Ausführliche Beschreibung . . . . .	51
5.35.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	51
5.35.3 Dokumentation der Elementfunktionen . . . . .	51
5.36 ComObjects.ComObject Schnittstellenreferenz . . . . .	51
5.36.1 Ausführliche Beschreibung . . . . .	52
5.36.2 Dokumentation der Elementfunktionen . . . . .	52
5.37 ComObjects.ComRuleset Klassenreferenz . . . . .	52
5.37.1 Ausführliche Beschreibung . . . . .	53

5.37.2	Beschreibung der Konstruktoren und Destruktoren	53
5.37.3	Dokumentation der Elementfunktionen	53
5.38	ComObjects.ComServerAcknowledgement Klassenreferenz	53
5.38.1	Ausführliche Beschreibung	54
5.38.2	Dokumentation der Elementfunktionen	54
5.39	ComObjects.ComStartGame Klassenreferenz	54
5.39.1	Ausführliche Beschreibung	54
5.39.2	Dokumentation der Elementfunktionen	54
5.40	ComObjects.ComUpdatePlayerlist Klassenreferenz	55
5.40.1	Ausführliche Beschreibung	55
5.40.2	Beschreibung der Konstruktoren und Destruktoren	55
5.40.3	Dokumentation der Elementfunktionen	55
5.41	ComObjects.ComWarning Klassenreferenz	56
5.41.1	Ausführliche Beschreibung	56
5.41.2	Beschreibung der Konstruktoren und Destruktoren	57
5.41.3	Dokumentation der Elementfunktionen	57
5.42	ComObjects.MsgCard Klassenreferenz	57
5.42.1	Ausführliche Beschreibung	58
5.42.2	Beschreibung der Konstruktoren und Destruktoren	58
5.42.3	Dokumentation der Elementfunktionen	58
5.43	ComObjects.MsgCardRequest Klassenreferenz	58
5.43.1	Ausführliche Beschreibung	59
5.43.2	Dokumentation der Elementfunktionen	59
5.44	ComObjects.MsgGameEnd Klassenreferenz	59
5.44.1	Ausführliche Beschreibung	59
5.44.2	Beschreibung der Konstruktoren und Destruktoren	60
5.44.3	Dokumentation der Elementfunktionen	61
5.45	ComObjects.MsgMultiCards Klassenreferenz	61
5.45.1	Ausführliche Beschreibung	61
5.45.2	Beschreibung der Konstruktoren und Destruktoren	62
5.45.3	Dokumentation der Elementfunktionen	62
5.46	ComObjects.MsgMultiCardsRequest Klassenreferenz	62
5.46.1	Ausführliche Beschreibung	63
5.46.2	Beschreibung der Konstruktoren und Destruktoren	63
5.46.3	Dokumentation der Elementfunktionen	63
5.47	ComObjects.MsgNumber Klassenreferenz	63
5.47.1	Ausführliche Beschreibung	64
5.47.2	Beschreibung der Konstruktoren und Destruktoren	64
5.47.3	Dokumentation der Elementfunktionen	64
5.48	ComObjects.MsgNumberRequest Klassenreferenz	65

5.48.1 Ausführliche Beschreibung . . . . .	65
5.48.2 Dokumentation der Elementfunktionen . . . . .	65
5.49 ComObjects.MsgSelection Klassenreferenz . . . . .	65
5.49.1 Ausführliche Beschreibung . . . . .	66
5.49.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	66
5.49.3 Dokumentation der Elementfunktionen . . . . .	66
5.50 ComObjects.MsgSelectionRequest Klassenreferenz . . . . .	66
5.50.1 Ausführliche Beschreibung . . . . .	67
5.50.2 Dokumentation der Elementfunktionen . . . . .	67
5.51 ComObjects.MsgUser Klassenreferenz . . . . .	67
5.51.1 Ausführliche Beschreibung . . . . .	67
5.51.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	67
5.51.3 Dokumentation der Elementfunktionen . . . . .	68
5.52 ComObjects.RulesetMessage Schnittstellenreferenz . . . . .	68
5.52.1 Ausführliche Beschreibung . . . . .	68
5.52.2 Dokumentation der Elementfunktionen . . . . .	69
5.53 Ruleset.Card Schnittstellenreferenz . . . . .	70
5.53.1 Ausführliche Beschreibung . . . . .	70
5.53.2 Dokumentation der Elementfunktionen . . . . .	70
5.54 Ruleset.ClientHearts Klassenreferenz . . . . .	71
5.54.1 Ausführliche Beschreibung . . . . .	71
5.54.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	71
5.54.3 Dokumentation der Elementfunktionen . . . . .	71
5.55 Ruleset.ClientRuleset Klassenreferenz . . . . .	72
5.55.1 Ausführliche Beschreibung . . . . .	73
5.55.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	73
5.55.3 Dokumentation der Elementfunktionen . . . . .	73
5.56 Ruleset.ClientWizard Klassenreferenz . . . . .	76
5.56.1 Ausführliche Beschreibung . . . . .	76
5.56.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	76
5.56.3 Dokumentation der Elementfunktionen . . . . .	76
5.57 Ruleset.Colour Enum-Referenz . . . . .	77
5.57.1 Ausführliche Beschreibung . . . . .	77
5.58 Ruleset.GameClientUpdate Klassenreferenz . . . . .	77
5.58.1 Ausführliche Beschreibung . . . . .	78
5.58.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	78
5.58.3 Dokumentation der Elementfunktionen . . . . .	78
5.59 Ruleset.GamePhase Enum-Referenz . . . . .	79
5.59.1 Ausführliche Beschreibung . . . . .	80
5.60 Ruleset.GameState Klassenreferenz . . . . .	80

5.60.1 Ausführliche Beschreibung . . . . .	80
5.60.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	80
5.60.3 Dokumentation der Elementfunktionen . . . . .	81
5.61 Ruleset.HeartsCard Enum-Referenz . . . . .	84
5.61.1 Ausführliche Beschreibung . . . . .	84
5.61.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	84
5.61.3 Dokumentation der Elementfunktionen . . . . .	84
5.62 Ruleset.HeartsData Klassenreferenz . . . . .	85
5.62.1 Ausführliche Beschreibung . . . . .	85
5.63 Ruleset.OtherData Klassenreferenz . . . . .	85
5.63.1 Ausführliche Beschreibung . . . . .	85
5.63.2 Dokumentation der Elementfunktionen . . . . .	86
5.64 Ruleset.PlayerState Klassenreferenz . . . . .	86
5.64.1 Ausführliche Beschreibung . . . . .	87
5.64.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	87
5.64.3 Dokumentation der Elementfunktionen . . . . .	87
5.65 Ruleset.RulesetType Enum-Referenz . . . . .	88
5.65.1 Ausführliche Beschreibung . . . . .	88
5.66 Ruleset.ServerHearts Klassenreferenz . . . . .	88
5.66.1 Ausführliche Beschreibung . . . . .	89
5.66.2 Dokumentation der Elementfunktionen . . . . .	89
5.67 Ruleset.ServerRuleset Klassenreferenz . . . . .	90
5.67.1 Ausführliche Beschreibung . . . . .	91
5.67.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	91
5.67.3 Dokumentation der Elementfunktionen . . . . .	91
5.68 Ruleset.ServerWizard Klassenreferenz . . . . .	96
5.68.1 Ausführliche Beschreibung . . . . .	97
5.68.2 Dokumentation der Elementfunktionen . . . . .	97
5.68.3 Dokumentation der Datenelemente . . . . .	98
5.69 Ruleset.WizardCard Enum-Referenz . . . . .	99
5.69.1 Ausführliche Beschreibung . . . . .	99
5.69.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	99
5.69.3 Dokumentation der Elementfunktionen . . . . .	99
5.70 Ruleset.WizData Klassenreferenz . . . . .	100
5.70.1 Ausführliche Beschreibung . . . . .	100
5.70.2 Dokumentation der Elementfunktionen . . . . .	100
5.71 Server.GameServer Klassenreferenz . . . . .	100
5.71.1 Ausführliche Beschreibung . . . . .	101
5.71.2 Beschreibung der Konstruktoren und Destrukturen . . . . .	101
5.71.3 Dokumentation der Elementfunktionen . . . . .	102



5.71.4	Dokumentation der Datenelemente . . . . .	104
5.72	Server.GameServerRepresentation Klassenreferenz . . . . .	104
5.72.1	Ausführliche Beschreibung . . . . .	105
5.72.2	Beschreibung der Konstruktoren und Destruktoren . . . . .	105
5.73	Server.LobbyServer Klassenreferenz . . . . .	105
5.73.1	Ausführliche Beschreibung . . . . .	106
5.73.2	Dokumentation der Elementfunktionen . . . . .	106
5.74	Server.LobbyServer.ClientListenerThread Klassenreferenz . . . . .	108
5.74.1	Ausführliche Beschreibung . . . . .	108
5.75	Server.Player Klassenreferenz . . . . .	108
5.75.1	Ausführliche Beschreibung . . . . .	109
5.75.2	Beschreibung der Konstruktoren und Destruktoren . . . . .	109
5.75.3	Dokumentation der Elementfunktionen . . . . .	109
5.76	Server.Server Klassenreferenz . . . . .	110
5.76.1	Ausführliche Beschreibung . . . . .	110
5.76.2	Dokumentation der Elementfunktionen . . . . .	110
5.77	Server.ServerMain Klassenreferenz . . . . .	111
5.77.1	Ausführliche Beschreibung . . . . .	112
5.77.2	Dokumentation der Elementfunktionen . . . . .	112
<b>6</b>	<b>JUnit-Tests</b>	<b>112</b>
6.1	isValidWizardMove . . . . .	112
6.2	isValidHeartsMove . . . . .	114
6.3	getWinner . . . . .	115
6.4	QuitPlayer . . . . .	119
6.5	Chat . . . . .	121
<b>7</b>	<b>Implementierungsplan</b>	<b>123</b>
7.1	Milestone 1 . . . . .	123
7.2	Milestone 2 . . . . .	124
7.3	Milestone 3 . . . . .	125
7.4	Finale Version . . . . .	125
7.5	Gantt-Diagramme . . . . .	126
<b>Index</b>		<b>128</b>

## 1 Einleitung

In diesem Dokument werden spezifische Angaben zu den bereits im Entwurf vorgestellten Klassen gemacht. Es werden im Folgenden Methoden und Attribute, sowie die Klassen genau beschrieben. Zunächst werden dazu die Änderungen vorgestellt, die seit dem Entwurf vorgenommen wurden. Des weiteren werden JUnit-Tests gezeigt,

die zum Testen des späteren Programms essentiell sind. Zuletzt wird ein Implementierungsplan mit verschiedenen Milestones aufgezeigt.

## 2 Änderungen im Vergleich zum Entwurf

### 2.1 Server

#### 2.1.1 Server

Die Klasse Server war zuvor ein Interface und ist jetzt eine abstrakte Klasse von der LobbyServer und der GameServer erben. So kann Codeduplikation vermieden werden.

### 2.2 Client

#### 2.2.1 Language

Das Enum Language wurde in das Packet ClientView verschoben, da es von der View benötigt wird um die richtige Sprache anzuzeigen.

#### 2.2.2 MVMessage

Die MVMessages wurden gelöscht und durch ein Enum ViewNotification ersetzt. Der Observer bekommt Änderungen über die Notifications mitgeteilt. Dies macht eine weitere Einteilung in einzelne Messages unnötig.

#### 2.2.3 ClientState-Enum

Dieses Enum enthält jeden Status, den ein Client erreichen kann.

### 2.3 ClientView

#### 2.3.1 MVMessages

Dieses Interface wurde gelöscht, da es unnötig ist.

#### 2.3.2 ViewCard

ViewCard ist die View-seitige Repräsentation einer Karte. Sie wird verwendet um einzelne Karten auf das Spielfeld zu zeichnen.

### 2.4 Ruleset

#### 2.4.1 ServerRuleset

Das ServerRuleset ist statt einem Interface nun eine abstrakte Klasse, von der die Klassen ServerHearts (vorher Hearts) und ServerWizard (vorher Wizard) erben. Codeduplikation kann so vermieden werden.

### 2.4.2 ClientRuleset

Das ClientRuleset ist statt einem Interface nun eine abstrakte Klasse, von der die Klassen ClientWizard und ClientHearts erben. So kann Codeduplikation vermieden werden.

### 2.4.3 Card

Die vorher abstrakte Klasse Card ist nun ein Interface, das von HeartsCard und WizardCard implementiert wird. So können die Methoden der Klasse Card von den anderen beiden so implementiert werden, wie es für das Spiel nötig ist.

### 2.4.4 Colour

Ein Enum Colour wurde hinzugefügt. Dieses enthält alle Farben, die eine Karte haben kann unabhängig vom Regelwerk.

### 2.4.5 RulesetType

Ein Enum RulesetType wurde hinzugefügt. Dieses enthält die Typen der Regelwerke, also deren Namen.

## 2.5 ComObjects

### 2.5.1 ComObject

ComObject ist im Zuge des Einsetzens des VisitorPatterns von einer abstrakten Klassen zu einem Interface geändert worden.

### 2.5.2 RulesetMessage

RulesetMessage ist im Zuge des Einsetzens des VisitorPatterns von einer abstrakten Klassen zu einem Interface geändert worden.

### 2.5.3 ComLogin

Diese Klassen trägt den neuen Namen ComLoginRequest.

### 2.5.4 ComKickPlayer

Diese Klasse trägt den neuen Namen ComKickPlayerRequest.

### 2.5.5 GameClientUpdate

Diese Klasse wurde gelöscht, da sie unnötig ist.

### 2.5.6 commands

Dieses Enum wurde gelöscht, da es unnötig ist. An Stelle von diesem wurden weitere ComObjects hinzugefügt.

### 2.5.7 types

Dieses Enum wurde gelöscht, da es unnötig ist.

#### 2.5.8 ComBeenKicked

Diese Klasse wurde zusätzlich hinzugefügt. Die Nachricht wird an einen Spieler gesendet, wenn er aus einem Spiel entfernt wurde.

#### 2.5.9 ComClientLeave

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

#### 2.5.10 ComClientQuit

Diese Klasse wurde zusätzlich hinzugefügt. Die Nachricht wird verschickt, wenn der Spieler ein Fenster schließt.

#### 2.5.11 ComCreateGameRequest

Diese Klasse wurde zusätzlich hinzugefügt. Diese Nachricht wird versendet, wenn ein neues Spiel erstellt werden soll.

#### 2.5.12 ComServerAcknowledgement

Diese Klasse wurde zusätzlich hinzugefügt. Diese Nachricht wird vom Server zur Bestätigung gesendet.

#### 2.5.13 ComStartGame

Diese Klasse wurde zusätzlich hinzugefügt. Diese Nachricht wird versendet, wenn ein Spiel gestartet werden soll.

#### 2.5.14 ComWarning

Diese Klasse wurde zusätzlich hinzugefügt. Diese Nachricht wird versendet, wenn ein Fehler passiert ist.

#### 2.5.15 MsgCardRequest

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

#### 2.5.16 MsgMultiCardsRequest

Diese Klasse wurde zusätzlich hinzugefügt. Diese Nachricht wird gesendet, wenn die Auswahl mehrerer Karten vom Spieler gefordert werden soll.

#### 2.5.17 MsgNumberRequest

Diese Klasse wurde zusätzlich hinzugefügt. Diese Nachricht wird gesendet, wenn die Eingabe einer Zahl gefordert werden soll.

#### 2.5.18 MsgSelectionRequest

Diese Klasse wurde zusätzlich hinzugefügt. Diese Nachricht sendet der Server an einen Spieler, wenn er eine Farbauswahl von diesem erwartet.

## 2.5.19 ComObjects-Klassen allgemein

Da die ehemals abstrakten Klassen ComObject und RulesetMessage jetzt Interfaces sind, implementieren alle anderen Objekte diese statt von ihnen zu erben, des Weiteren implementieren alle Serializable.

## 3 Hierarchie-Verzeichnis

### 3.1 Klassenhierarchie

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

<b>Client.ClientController</b>	<b>11</b>
<b>Client.ClientMain</b>	<b>12</b>
<b>Client.ClientModel</b>	<b>12</b>
<b>Client.ClientState</b>	<b>20</b>
<b>Client.MessageListenerThread</b>	<b>21</b>
<b>Client.View.ChooseCards</b>	<b>21</b>
<b>Client.View.ChooseItem</b>	<b>22</b>
<b>Client.View.CreateGame</b>	<b>23</b>
<b>Client.View.DiscardPile</b>	<b>24</b>
<b>Client.View.DrawDeck</b>	<b>24</b>
<b>Client.View.Game</b>	<b>24</b>
<b>Client.View.GameLobby</b>	<b>26</b>
<b>Client.View.GamePanel</b>	<b>28</b>
<b>Client.View.InputNumber</b>	<b>28</b>
<b>Client.View.Language</b>	<b>29</b>
<b>Client.View.Lobby</b>	<b>29</b>
<b>Client.View.Login</b>	<b>31</b>
<b>Client.View.OtherPlayer</b>	<b>33</b>
<b>Client.View.OwnHand</b>	<b>33</b>
<b>Client.View.Password</b>	<b>33</b>
<b>Client.View.ScoreWindow</b>	<b>34</b>
<b>Client.View.ViewCard</b>	<b>35</b>
<b>Client.View.Warning</b>	<b>36</b>
<b>Client.ViewNotification</b>	<b>37</b>
<b>ComObjects.ComObject</b>	<b>51</b>

ComObjects.ComBeenKicked	37
ComObjects.ComChatMessage	38
ComObjects.ComClientLeave	39
ComObjects.ComClientQuit	40
ComObjects.ComCreateGameRequest	41
ComObjects.ComInitGameLobby	43
ComObjects.ComInitLobby	44
ComObjects.ComJoinRequest	45
ComObjects.ComKickPlayerRequest	47
ComObjects.ComLobbyUpdateGamelist	49
ComObjects.ComLoginRequest	50
ComObjects.ComRuleset	52
ComObjects.ComServerAcknowledgement	53
ComObjects.ComStartGame	54
ComObjects.ComUpdatePlayerlist	55
ComObjects.ComWarning	56
ComObjects.RulesetMessage	68
ComObjects.MsgCard	57
ComObjects.MsgCardRequest	58
ComObjects.MsgGameEnd	59
ComObjects.MsgMultiCards	61
ComObjects.MsgMultiCardsRequest	62
ComObjects.MsgNumber	63
ComObjects.MsgNumberRequest	65
ComObjects.MsgSelection	65
ComObjects.MsgSelectionRequest	66
ComObjects.MsgUser	67
Ruleset.Card	70
Ruleset.HeartsCard	84
Ruleset.WizardCard	99
Ruleset.ClientRuleset	72
Ruleset.ClientHearts	71

Ruleset.ClientWizard	76
Ruleset.Colour	77
Ruleset.GameClientUpdate	77
Ruleset.GamePhase	79
Ruleset.GameState	80
Ruleset.OtherData	85
Ruleset.HeartsData	85
Ruleset.WizData	100
Ruleset.PlayerState	86
Ruleset.RulesetType	88
Ruleset.ServerRuleset	90
Ruleset.ServerHearts	88
Ruleset.ServerWizard	96
Server.GameServerRepresentation	104
Server.LobbyServer.ClientListenerThread	108
Server.Player	108
Server.Server	110
Server.GameServer	100
Server.LobbyServer	105
Server.ServerMain	111

## 4 Klassen-Verzeichnis

### 4.1 Auflistung der Klassen

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

Client.ClientController	
ClientController	11
Client.ClientMain	
ClientMain	12
Client.ClientModel	
ClientModel	12
Client.ClientState	
ClientState	20
Client.MessageListenerThread	
MessageListenerThread	21

Client.View.ChooseCards ChooseCards	21
Client.View.ChooseItem ChooseItem	22
Client.View.CreateGame CreateGame	23
Client.View.DiscardPile DiscardPile	24
Client.View.DrawDeck DrawDeck	24
Client.View.Game Game	24
Client.View.GameLobby GameLobby	26
Client.View.GamePanel GamePanel	28
Client.View.InputNumber InputNumber	28
Client.View.Language Language	29
Client.View.Lobby Lobby	29
Client.View.Login Login	31
Client.View.OtherPlayer OtherPlayer	33
Client.View.OwnHand OwnHand	33
Client.View.Password Password	33
Client.View.ScoreWindow ScoreWindow	34
Client.View.ViewCard ViewCard	35
Client.View.Warning Warning	36
Client.View.Notification ViewNotification	37
ComObjects.ComBeenKicked ComBeenKicked	37
ComObjects.ComChatMessage ComChatMessage	38



ComObjects.ComClientLeave ComClientLeave	39
ComObjects.ComClientQuit ComClientQuit	40
ComObjects.ComCreateGameRequest ComCreateGameRequest	41
ComObjects.ComInitGameLobby ComInitGameLobby	43
ComObjects.ComInitLobby ComInitLobby	44
ComObjects.ComJoinRequest ComJoinRequest	45
ComObjects.ComKickPlayerRequest ComKickPlayerRequest	47
ComObjects.ComLobbyUpdateGamelist ComLobbyUpdateGamelist	49
ComObjects.ComLoginRequest ComLoginRequest	50
ComObjects.ComObject ComObject	51
ComObjects.ComRuleset ComRuleset	52
ComObjects.ComServerAcknowledgement ComServerAcknowledgement	53
ComObjects.ComStartGame ComStartGame	54
ComObjects.ComUpdatePlayerlist ComUpdatePlayerlist	55
ComObjects.ComWarning ComWarning	56
ComObjects.MsgCard MsgCard	57
ComObjects.MsgCardRequest MsgCardRequest	58
ComObjects.MsgGameEnd MsgGameEnd	59
ComObjects.MsgMultiCards MsgMultiCards	61
ComObjects.MsgMultiCardsRequest MsgMultiCardsRequest	62
ComObjects.MsgNumber MsgNumber	63

ComObjects.MsgNumberRequest MsgNumberRequest	65
ComObjects.MsgSelection MsgSelection	65
ComObjects.MsgSelectionRequest MsgSelectionRequest	66
ComObjects.MsgUser MsgUser	67
ComObjects.RulesetMessage RulesetMessage	68
Ruleset.Card Card	70
Ruleset.ClientHearts ClientHearts	71
Ruleset.ClientRuleset ClientRuleset	72
Ruleset.ClientWizard ClientWizard	76
Ruleset.Colour Colour	77
Ruleset.GameClientUpdate GameClientUpdate	77
Ruleset.GamePhase GamePhase	79
Ruleset.GameState GameState	80
Ruleset.HeartsCard HeartsCard	84
Ruleset.HeartsData HeartsData	85
Ruleset.OtherData OtherData	85
Ruleset.PlayerState PlayerState	86
Ruleset.RulesetType Ruleset	88
Ruleset.ServerHearts ServerHearts	88
Ruleset.ServerRuleset ServerRuleset	90
Ruleset.ServerWizard ServerWizard	96

<a href="#">Ruleset.WizardCard</a> <a href="#">WizardCard</a>	99
<a href="#">Ruleset.WizData</a> <a href="#">WizData</a>	100
<a href="#">Server.GameServer</a> <a href="#">GameServer</a>	100
<a href="#">Server.GameServerRepresentation</a> <a href="#">GameServerRepresentation</a>	104
<a href="#">Server.LobbyServer</a> <a href="#">LobbyServer</a>	105
<a href="#">Server.LobbyServer.ClientListenerThread</a> <a href="#">ClientListenerThread</a>	108
<a href="#">Server.Player</a> <a href="#">Player</a>	108
<a href="#">Server.Server</a> <a href="#">Server</a>	110
<a href="#">Server.ServerMain</a> <a href="#">ServerMain</a>	111

## 5 Klassen-Dokumentation

### 5.1 Client.ClientController Klassenreferenz

#### Private Attribute

- [ClientModel](#) **clientModel**
- [ChooseCards](#) **chooseCards**
- [ScoreWindow](#) **scoreWindow**
- [InputNumber](#) **inputNumber**
- [ChooseItem](#) **chooseColour**
- [Lobby](#) **lobby**
- [Set< Login >](#) **login**
- [CreateGame](#) **createGame**
- [GameLobby](#) **gameLobby**
- [Game](#) **game**
- [Password](#) **password**
- [Language](#) **language**
- [Set< Warning >](#) **warning**

#### 5.1.1 Ausführliche Beschreibung

Der [ClientController](#) enthaelt alle ActionListener der View und leitet durch diese Benutzereingaben an das [Client-Model](#) weiter. Sobald der [ClientController](#) von der ClientMain-Klasse erzeugt wird, erzeugt er wiederum das [Client-Model](#) und die Fenster der ClientView, wobei zunaechst nur das Login Fenster sichtbar ist.

## 5.2 Client.ClientMain Klassenreferenz

### Öffentliche, statische Methoden

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

### Private Attribute

- [ClientController](#) [clientController](#)

### 5.2.1 Ausführliche Beschreibung

Die [ClientMain](#) Klasse startet den Spielclient und initialisiert dessen Komponenten.

### 5.2.2 Dokumentation der Elementfunktionen

#### 5.2.2.1 static void Client.ClientMain.main ( final String[] args ) [static]

#### Parameter

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

## 5.3 Client.ClientModel Klassenreferenz

Abgeleitet von Observable.

### Öffentliche Methoden

- [ClientModel](#) ([MessageListenerThread](#) netIO) throws [IllegalArgumentException](#)
- void [leaveWindow](#) ()
- void [terminateProgram](#) ()
- 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](#) com)
- List< String > [getPlayerlist](#) ()
- List< [GameServerRepresentation](#) > [getLobbyGamelist](#) ()
- List< [Card](#) > [getPlayedCards](#) ()
- List< [Card](#) > [getOwnHand](#) ()
- List< String > [getOtherPlayerData](#) ()
- int [getOwnScore](#) ()
- void [setLanguage](#) (final [Language](#) language)
- [Language](#) [getLanguage](#) ()
- void [kickPlayer](#) (final String name)
- void [hostGame](#) (String gameName, boolean hasPassword, String password, [RulesetType](#) ruleset)
- void [send](#) ([ComObject](#) object)
- void [send](#) ([RulesetMessage](#) msg)
- int [getPlayerCount](#) ()

- String [getWindowText](#) ()
- List< [Card](#) > [getChooseCards](#) ()
- void [giveChosenCards](#) (List< [Card](#) > cards)
- void [openChooseCards](#) (List< [Card](#) > cards, String text)
- List< Object > [getChooselItems](#) ()
- void [giveChosenItem](#) (Object item)
- void [openChooselItem](#) (List< Object > items, String text)
- void [giveInputNumber](#) (int number)
- void [openInputNumber](#) (String text)
- void [sendChatMessage](#) (final String msg)
- void [joinGame](#) (final String name, final String password)
- void [startGame](#) ()
- void [makeMove](#) ([Card](#) card)
- void [announceWinner](#) (final String winner)
- String [getWinner](#) ()
- void [createConnection](#) (final String username, final String serverAdress, final int port)
- String [getWarningText](#) ()
- List< [RulesetType](#) > [getRulesets](#) ()

#### Private Methoden

- void [initGame](#) ()
- void [informView](#) ([ViewNotification](#) note)

#### Private Attribute

- String [playerName](#)
- [ClientRuleset](#) ruleset
- [Language](#) language
- [ClientState](#) state
- List< String > **playerList**
- String **chatMessage**
- Set< [GameServerRepresentation](#) > **gameList**
- [MessageListenerThread](#) netIO

#### 5.3.1 Ausführliche Beschreibung

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

#### 5.3.2 Beschreibung der Konstruktoren und Destruktoren

##### 5.3.2.1 Client.ClientModel.ClientModel ( [MessageListenerThread](#) netIO ) throws [IllegalArgumentException](#)

Erstellt ein [ClientModel](#) und erwartet als Argument einen [MessageListenerThread](#) fuer die Netzwerkanbindung.

**Parameter**

<i>netIO</i>	<a href="#">MessageListenerThread</a> fuer die Netzwerkverbindung.
--------------	--

**Ausnahmebehandlung**

<i>IllegalArgumentException</i>	Wird geworfen bei fehlerhaftem <a href="#">MessageListenerThread</a> Argument.
---------------------------------	--

Benutzt Client.ClientModel.netIO.

**5.3.3 Dokumentation der Elementfunktionen****5.3.3.1 void Client.ClientModel.leaveWindow ( )**

Wird aufgerufen, wenn der User die GameLobby verlaesst.

Der Client gelangt zurueck in die Lobby.

**5.3.3.2 void Client.ClientModel.receiveMessage ( ComChatMessage msg )**

Sendet eine eingehende Chatnachricht direkt an alle Observer weiter.

**Parameter**

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

**5.3.3.3 void Client.ClientModel.receiveMessage ( ComInitLobby msg )**

Diese Methode wird aufgerufen, falls der Server den Spieler erfolgreich in die Lobby hinzugefügt hat.

Empfaengt die ComInitGameLobby Nachricht, die eine Liste aller Spieler enthaelt, die sich in der Lobby befinden. Speichert diese Liste und benachrichtigt die Observer mit der loginSuccesful [ViewNotification](#).

**Parameter**

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

**5.3.3.4 void Client.ClientModel.receiveMessage ( ComInitGameLobby msg )**

Diese Methode wird aufgerufen, falls der Server den Spieler erfolgreich in die GameLobby hinzugefuegt hat.

Empfaengt die ComInitGameLobby Nachricht, die eine Liste aller Spieler enthaelt, die sich in der GameLobby befinden. Speichert diese Liste und benachrichtigt die Observer mit der joinGameSuccesful [ViewNotification](#).

**Parameter**

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

**5.3.3.5 void Client.ClientModel.receiveMessage ( ComRuleset msg )**

Diese Methode wird aufgerufen, falls eine Nachricht für das Regelwerk ankommt.

Die darin enthaltene RulesetMessage wird dem ClientRuleset zur Verarbeitung uebergeben.

**Parameter**

<i>msg</i>	Die ankommende ComRuleset Nachricht
------------	-------------------------------------

**5.3.3.6 void Client.ClientModel.receiveMessage ( ComServerAcknowledgement ack )**

Diese Methode wird 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.
------------	------------------------------------

## 5.3.3.7 void Client.ClientModel.receiveMessage ( ComBeenKicked msg )

Diese Methode wird aufgerufen, falls der Spieler aus der Spiellobby durch einen Spielleiter entfernt wurde.

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

## Parameter

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

## 5.3.3.8 void Client.ClientModel.receiveMessage ( ComUpdatePlayerlist update )

Diese Methode wird aufgerufen, falls auf dem Server ein neuer Spieler die Lobby/GameLobby betreten hat oder sie von einem Spieler verlassen wurde.

Empfaengt die ComUpdatePlayerlist Nachricht, die die Information enthaelt, ob und welcher Spieler hinzugefuegt oder entfernt werden muss. Die Spielerliste wird dementsprechend bearbeitet und die Observer mit playerList-Update informiert.

## Parameter

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

## 5.3.3.9 void Client.ClientModel.receiveMessage ( ComLobbyUpdateGamelist update )

Diese Methode wird aufgerufen, falls auf dem Server ein neues Spiel erstellt wurde oder ein Spiel geschlossen/beendet wurde.

Empfaengt die ComLobbyUpdateGamelist Nachricht, die die Information enthaelt, ob und welches Spiel hinzugefuegt oder entfernt werden muss. Die Spielliste wird dementsprechend bearbeitet und die Observer mit gameList-Update informiert.

## Parameter

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

## 5.3.3.10 void Client.ClientModel.receiveMessage ( ComObject com )

Standard receiveMessage Methode, die ComObjekte zur Weiterverarbeitung identifiziert.

## Parameter

<i>com</i>	Das auflaufende ComObjekt.
------------	----------------------------

## 5.3.3.11 List&lt;String&gt; Client.ClientModel.getPlayerlist ( )

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

## Rückgabe

Liste von Spielernamen

## 5.3.3.12 List&lt;GameServerRepresentation&gt; Client.ClientModel.getLobbyGamelist ( )

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

**Rückgabe**

Liste aller Spiele der Lobby.

**5.3.3.13 List<Card> Client.ClientModel.getPlayedCards ( )**

Gibt eine Liste aller bereits ausgespielten Karten zurueck.

**Rückgabe**

List<Card>. Eine Liste der gespielten Karten.

**5.3.3.14 List<Card> Client.ClientModel.getOwnHand ( )**

Gibt eine Liste der Handkarten des Spielers zurueck.

**Parameter**

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

**5.3.3.15 List<String> Client.ClientModel.getOtherPlayerData ( )**

Gibt zusaetzliche Daten der anderen Spieler zurueck.

**Rückgabe**

Liste der Stringrepraesentationen der OtherData der Spieler

**5.3.3.16 int Client.ClientModel.getOwnScore ( )**

Gibt den Punktestand des Spielers zurueck.

**Rückgabe**

der eigene Punktestand.

**5.3.3.17 void Client.ClientModel.setLanguage ( final Language language )**

Setzt die Sprache der GUI.

**Parameter**

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

Benutzt Client.ClientModel.language.

**5.3.3.18 Language Client.ClientModel.getLanguage ( )**

Liefert die Sprache der GUI.

**Rückgabe**

language Enumerator der die Spielsprache anzeigt.

Benutzt Client.ClientModel.language.

**5.3.3.19 void Client.ClientModel.kickPlayer ( final String name )**

Entfernt einen Spieler aus der GameLobby.



## Parameter

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

**5.3.3.20** void Client.ClientModel.hostGame ( String *gameName*, boolean *hasPassword*, String *password*, RulesetType *ruleset* )

Erstellt ein neues Spiel.

Sendet dazu eine ComCreateGameRequest Nachricht an den Server.

## Parameter

<i>gameName</i>	String Name des Spieles.
<i>hasPassword</i>	true, wenn das Spiel ein Passwort hat
<i>password</i>	String Passwort zum sichern des Spieles.
<i>ruleset</i>	das zu verwendende Regelwerk

**5.3.3.21** void Client.ClientModel.send ( ComObject *object* )

Sendet erstellte ComObjects an den Server.

## Parameter

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

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

**5.3.3.22** void Client.ClientModel.send ( RulesetMessage *msg* )

Sendet eine RulesetMessage an den Server.

Erstellt dazu eine ComRuleset, die die RulesetMessage enthaelt.

## Parameter

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

Benutzt Client.ClientModel.send().

**5.3.3.23** int Client.ClientModel.getPlayerCount ( )

Gibt die Anzahl der Spieler eines Spieles zurueck.

## Rückgabe

int Die Spielerzahl eines Spieles.

**5.3.3.24** String Client.ClientModel.getWindowText ( )

Gibt den Text zurueck, der in einem Sonderfenster (InputNumber, ChooseItem, ChooseCards) angezeigt werden soll.

## Rückgabe

String

**5.3.3.25** List<Card> Client.ClientModel.getChooseCards ( )

Gibt die Karten zurueck, aus denen gewaehlt werden soll.

## Rückgabe

Karten, aus denen gewaehlt werden kann

### 5.3.3.26 void Client.ClientModel.giveChosenCards ( List< Card > cards )

Uebergibt die Karten, die vom User gewahlt wurden.

Diese werden dann dem Regelwerk weitergegeben. Akzeptiert dieses die gewaehlten Karten nicht, wird nochmal openChooseCards aufgerufen.

Parameter

<i>cards</i>	Karten, die der User gewaehlt hat
--------------	-----------------------------------

### 5.3.3.27 void Client.ClientModel.openChooseCards ( List< Card > cards, String text )

Benachrichtigt die Observer mit der openChooseCards [ViewNotification](#) und speichert die Liste der Karten sowie den Anzeigetext des Regelwerks zwischen.

Parameter

<i>cards</i>	Liste der Karten, von denen gewaehlt werden kann
<i>text</i>	Text, der dem User angezeigt werden soll

### 5.3.3.28 List<Object> Client.ClientModel.getChooseltems ( )

Gibt die Items zurueck, aus denen eines gewaehlt werden soll.

Rückgabe

Items, aus denen gewahlt werden kann

### 5.3.3.29 void Client.ClientModel.giveChosenItem ( Object item )

Uebergibt das Item, das vom User gewahlt wurden.

Dieses wird dann dem Regelwerk weitergegeben. Akzeptiert dieses das gewaehlte Item nicht, wird nochmal openChooseltem aufgerufen.

Parameter

<i>item</i>	Item, das der User gewaehlt hat
-------------	---------------------------------

### 5.3.3.30 void Client.ClientModel.openChooseltem ( List< Object > items, String text )

Benachrichtigt die Observer mit der openChooseltem [ViewNotification](#) und speichert die Liste der Items, von denen eines gewaehlt werden soll, sowie den Anzeigetext des Regelwerks zwischen.

Parameter

<i>items</i>	Liste der Items, von denen eines gewaehlt werden soll
<i>text</i>	Text, der dem User angezeigt werden soll

### 5.3.3.31 void Client.ClientModel.giveInputNumber ( int number )

Uebergibt die Zahl, die vom User gewahlt wurde.

Diese wird dann dem Regelwerk weitergegeben. Akzeptiert dieses die gewaehlte Zahl nicht, wird nochmal openInputNumber aufgerufen.

Parameter

<i>number</i>	Zahl, die vom User gewählt wurde
---------------	----------------------------------

#### 5.3.3.32 void Client.ClientModel.openInputNumber ( String text )

Benachrichtigt die Observer mit der openInputNumber [ViewNotification](#) und speichert den Anzeigetext des Regelwerks zwischen.

##### Parameter

<i>text</i>	Text, der dem User angezeigt werden soll
-------------	--

#### 5.3.3.33 void Client.ClientModel.sendChatMessage ( final String msg )

Nimmt vom [ClientController](#) eine Chatnachricht entgegen und sendet diese an den Server.

##### Parameter

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

Benutzt Client.ClientModel.send().

#### 5.3.3.34 void Client.ClientModel.joinGame ( final String name, final String password )

Versucht einem Spiel beizutreten.

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

##### Parameter

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

#### 5.3.3.35 void Client.ClientModel.startGame ( )

Versucht das erstellte Spiel zu starten.

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

#### 5.3.3.36 void Client.ClientModel.initGame ( ) [private]

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

Der Client gelangt ins Spiel Die Observer werden über die gameStarted [ViewNotification](#) benachrichtigt.

#### 5.3.3.37 void Client.ClientModel.makeMove ( Card card )

Versucht eine Karte auszuspielen.

Laesst dazu vom ClientRuleset ueberpruefen ob, die ausgewaehlte Karte gespielt werden darf. Wenn ja, wird sie im ClientRuleset weiterbehandelt. Wenn nein, wird eine Fehlermeldung ausgegeben und dazu die Observer mit openWarning informiert.

##### Parameter

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

#### 5.3.3.38 void Client.ClientModel.announceWinner ( final String winner )

Wird aufgerufen wenn das Ende einer Partie erreicht ist.

## Parameter

<i>winner</i>	String der Gewinner der Partie.
---------------	---------------------------------

## 5.3.3.39 String Client.ClientModel.getWinner ( )

Liefert den Gewinner einer Partie.

## Rückgabe

String der Gewinner.

## 5.3.3.40 void Client.ClientModel.informView ( ViewNotification note ) [private]

Hilfsmethode die alle verbundenen Observer der GUI kontaktiert.

## Parameter

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

## 5.3.3.41 void Client.ClientModel.createConnection ( final String username, final String serverAdress, final int port )

Erstellt den [MessageListenerThread](#) und fuehrt den Benutzerlogin durch.

## Parameter

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

## 5.3.3.42 String Client.ClientModel.getWarningText ( )

Gibt den Text aus der bei einer Spielwarnung angezeigt wird.

## Rückgabe

String Text der Warnung.

## 5.3.3.43 List&lt;RulesetType&gt; Client.ClientModel.getRulesets ( )

Liefert eine Liste mit allen implementierten Regelwerken.

## Parameter

<i>List&lt;RulesetType&gt;</i>	Liste von unterstuetzten Regelwerken.
--------------------------------	---------------------------------------

## 5.4 Client.ClientState Enum-Referenz

## Öffentliche Attribute

- [LOGIN](#)
- [SERVERLOBBY](#)
- [GAMECREATION](#)
- [PASSWORDREQUEST](#)
- [GAMELOBBY](#)
- [GAME](#)
- [USERREQUEST](#)
- [ENDING](#)

## 5.4.1 Ausführliche Beschreibung

Dieser Enumerator enthält alle Zustände in denen sich der Client befinden kann.

## 5.5 Client.MessageListenerThread Klassenreferenz

Abgeleitet von Thread.

## Öffentliche Methoden

- [MessageListenerThread](#) ()
- void [startConnection](#) ([ClientModel](#) model, Socket connection) throws IllegalArgumentException, IOException
- void [closeConnection](#) ()
- void [send](#) ([ComObject](#) object)
- void [run](#) ()

## Private Attribute

- Socket **socket**
- ObjectInput **in**
- ObjectOutput **out**
- boolean **run** = false
- [ClientModel](#) **model**

## 5.5.1 Ausführliche Beschreibung

Diese Klasse implementiert die Netzwerkanbindung des Clients an den Server. Sie enthält den dazu nötigen Socket und ObjectStream Reader und Writer.

## 5.5.2 Dokumentation der Elementfunktionen

5.5.2.1 void Client.MessageListenerThread.startConnection ( [ClientModel](#) model, Socket connection ) throws IllegalArgumentException, IOException

Initialisiert die ObjectStreams und speichert den TCP Socket im Thread.

## Parameter

<i>model</i>	<a href="#">ClientModel</a> , Das Model das den Spielablauf und Serverkommunikation steuert.
<i>connection</i>	Socket, der Socket über den die TCP Verbindung laeuft.

## Ausnahmebehandlung

<i>IllegalArgumentException</i>	Wird geworfen bei falschen <a href="#">ClientModel</a> oder Socket Argumenten.
<i>IOException</i>	Wird geworfen beim fehlerbehafteten Erstellen der ObjectStreams.

## 5.6 Client.View.ChooseCards Klassenreferenz

Abgeleitet von Observer.

## Öffentliche Methoden

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

## Private Attribute

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

### 5.6.1 Ausführliche Beschreibung

In diesem Fenster muss der Benutzer eine vorbestimmte Menge Karten auswaehlen.

### 5.6.2 Dokumentation der Elementfunktionen

#### 5.6.2.1 void Client.View.ChooseCards.update ( Observable o, Object arg )

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

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

#### Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<i>arg</i>	erwartet: openChooseCards, chooseCardsSuccessful

## 5.7 Client.View.Chooseltem Klassenreferenz

Abgeleitet von Observer.

## Öffentliche Methoden

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

## Private Attribute

- Object **itemComboBox**

### 5.7.1 Ausführliche Beschreibung

Dieses Fenster ermoeoglicht es dem Spieler aus einer Liste von Items eines auszuwaehlen.

### 5.7.2 Dokumentation der Elementfunktionen

#### 5.7.2.1 void Client.View.Chooseltem.update ( Observable arg0, Object arg1 )

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

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

#### Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<i>arg</i>	erwartet: openChooseltem, chooseltemSuccessful

## 5.8 Client.View.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

#### 5.8.1 Ausführliche Beschreibung

Das Fenster [CreateGame](#) dient dem Benutzer zur Erstellung eines neuen Spieles. Es bietet alle Komponenten, um ein Regelwerk zu waehlen, einen Spielnamen festzulegen und das Spiel durch ein Passwort zu schuetzen. In der Spielerstellung wird ein Titelbild des ausgewaehlten Spiels und eine kurze Beschreibung angezeigt. ueber 'Leave' kehrt der Spieler in die [Lobby](#) zurueck und mit 'Create' wird das Spiel erstellt.

#### 5.8.2 Dokumentation der Elementfunktionen

##### 5.8.2.1 void Client.View.CreateGame.addPanelMouseListener ( MouseListener m )

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

**Parameter**

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

**5.8.2.2 void Client.View.CreateGame.addRulesetSelectionListener ( ItemListener i )**

Fuegt einen Listener fuer die Regelwerk-Auswahl des `CreateGame` Fensters hinzu.

**Parameter**

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

**5.8.2.3 void Client.View.CreateGame.addCreateButtonListener ( ActionListener a )**

Fuegt einen ActionListener fuer den 'Create' Button hinzu.

**Parameter**

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

**5.8.2.4 void Client.View.CreateGame.addLeaveButtonListener ( ActionListener a )**

Fuegt einen ActionListener fuer den 'Leave' Button hinzu.

**Parameter**

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

**5.8.2.5 void Client.View.CreateGame.setLanguage ( Language l )**

Aendert die Sprache des Fensters.

**Parameter**

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

**5.9 Client.View.DiscardPile Klassenreferenz****Private Attribute**

- `Set< ViewCard > card`

**5.9.1 Ausführliche Beschreibung**

Stellt einen Ablagestapel dar, dieser kann sowohl für jeden Spieler einzeln oder für alle Spieler gemeinsam in der Mitte des Spielfeldes angezeigt werden.

**5.10 Client.View.DrawDeck Klassenreferenz****5.10.1 Ausführliche Beschreibung**

Stellt einen Aufnahmestapel dar.

**5.11 Client.View.Game Klassenreferenz**

Abgeleitet von `JFrame` und `Observer`.



## Öffentliche Methoden

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

## Öffentliche, statische Methoden

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

## Private Attribute

- JPanel **contentPane**
- JTextField **textField**
- [GamePanel](#) **panel**

## Statische, private Attribute

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

## 5.11.1 Ausführliche Beschreibung

Im [Game](#) Fenster laeuft das Spiel ab. Es enthaelt den Spielchat und ein [GamePanel](#). Ausserdem koennen ueber ein Dropdown-Menue Aenderungen an Hintergrundbild und Kartenhintergruenden vorgenommen werden. Schliesseen beendet das Spiel und der Spieler wird in die [Lobby](#) zurueckgeleitet.

## 5.11.2 Beschreibung der Konstruktoren und Destruktoren

## 5.11.2.1 Client.View.Game.Game ( ) throws IOException

Erstellt das [Game](#) Fenster.

## Ausnahmebehandlung

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

## 5.11.3 Dokumentation der Elementfunktionen

5.11.3.1 void Client.View.Game.makeTrickGameBoard ( int *playercount* )

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

Hierfuer hat jeder Spieler einen eigenen Ablagestapel vor sich. Es koennen 3, 4, 5, oder 6 Spieler gewaehlt werden.

## Parameter

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

5.11.3.2 void Client.View.Game.update ( Observable o, Object *arg* )

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

Je nach dem in arg uebergebenen ViewNotification-Befehl wird ein Update des Fensters ausgefuehrt 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

**5.11.3.3 void Client.View.Game.update ( Observable o, String arg )**

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

**Parameter**

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<i>arg</i>	erwartet eine Chatnachricht in String-Form

**5.12 Client.View.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

### 5.12.1 Ausführliche Beschreibung

Die [GameLobby](#) modelliert das Wartefenster, in dem beigetretene Spieler auf den Start des Spieles durch den Spielleiter warten. Der Spielleiter kann Spieler mit dem Remove Player Button entfernen. ueber Leave kehren die Spieler in die [Lobby](#) zurueck. Der spielinterne Chat ist ab hier verfuegbar.

### 5.12.2 Dokumentation der Elementfunktionen

#### 5.12.2.1 void Client.View.GameLobby.addStartButtonListener ( ActionListener a )

Fuegt einen ActionListener fuer den 'Start [Game](#)' Button hinzu.

Parameter

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

#### 5.12.2.2 void Client.View.GameLobby.addRemoveButtonListener ( ActionListener a )

Fuegt einen ActionListener fuer den 'Remove Player' Button hinzu.

Parameter

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

#### 5.12.2.3 void Client.View.GameLobby.addLeaveButtonListener ( ActionListener a )

Fuegt einen ActionListener fuer den 'Leave' Button hinzu.

Parameter

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

#### 5.12.2.4 void Client.View.GameLobby.addChatMessageListener ( KeyListener k )

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

Parameter

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

#### 5.12.2.5 void Client.View.GameLobby.setLanguage ( Language l )

Aendert die Sprache des Fensters.

Parameter

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

#### 5.12.2.6 void Client.View.GameLobby.update ( Observable o, Object arg )

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

Je nach dem in arg uebergebenen ViewNotification-Befehl wird ein Update des Fensters ausgefuehrt 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

#### 5.12.2.7 void Client.View.GameLobby.update ( Observable *o*, String *arg* )

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

Dieser wird als Chatnachricht interpretiert und dem Chatlog angefuegt.

##### Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<i>arg</i>	erwartet einen String, der eine Chatnachricht darstellt

### 5.13 Client.View.GamePanel Klassenreferenz

Abgeleitet von JPanel.

#### Öffentliche Methoden

- [GamePanel](#) ()
- void [makeTrickGameBoardThreePlayers](#) ()
- void [makeTrickGameBoardFourPlayers](#) ()
- void [makeTrickGameBoardFivePlayers](#) ()
- void [makeTrickGameBoardSixPlayers](#) ()
- 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

#### 5.13.1 Ausführliche Beschreibung

Das [GamePanel](#) ist die Komponente des Game-Fensters, welche das eigentliche Spiel darstellt. Es besteht aus verschiedenen Panelobjekten, welche je nach Regelwerk auf das Spielfeld gezeichnet werden. Dazu gehoeren die eigenen Karten, eventuell ausgewaehlte Karten, ein Textfeld z.B. zur Anzeige der Anzahl der restlichen Karten der Mitspieler und den Ablagestapel. Nach jeder Runde wird der Punktestand aktualisiert.

### 5.14 Client.View.InputNumber Klassenreferenz

Abgeleitet von Observer.

## Öffentliche Methoden

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

## Private Attribute

- Object **numberTextfield**

## 5.14.1 Ausführliche Beschreibung

In diesem Fenster, kann der Benutzer eine Zahl eingeben.

## 5.14.2 Dokumentation der Elementfunktionen

## 5.14.2.1 void Client.View.InputNumber.update ( Observable o, Object arg )

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

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

## Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<i>arg</i>	erwartet: openInputNumber, inputNumberSuccessful

## 5.15 Client.View.Language Enum-Referenz

## Öffentliche Attribute

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

## 5.15.1 Ausführliche Beschreibung

[Language](#) stellt Repraesentationen verschiedener Sprachen dar, die von der GUI verwendet werden, um festzustellen welche Anzeigesprache verwendet werden soll.

## 5.16 Client.View.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

## 5.16.1 Ausführliche Beschreibung

Diese Klasse erzeugt die Ansicht der ServerLobby auf der Client Seite, in der die Spieler neue Spiele erstellen oder offenen beitreten koennen. In der [Lobby](#) werden die Benutzernamen der sich in der [Lobby](#) befindenden Spieler, sowie offene Spiele angezeigt. In der [Lobby](#) koennen Chatnachrichten gesendet und empfangen werden. ueber 'Leave' verlaesst der Spieler das Spiel. ueber 'Host [Game](#)' wird der Spieler zum CreateGame-Fenster weiter geleitet und mit 'Join [Game](#)' kann einem bereits erstellten Spiel beigetreten werden.

## 5.16.2 Dokumentation der Elementfunktionen

## 5.16.2.1 void Client.View.Lobby.addJoinButtonListener ( ActionListener a )

Fuegt einen ActionListener fuer den 'Join' Button hinzu.

## Parameter

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

## 5.16.2.2 void Client.View.Lobby.addHostButtonListener ( ActionListener a )

Fuegt einen ActionListener fuer den 'Host' Button hinzu.

## Parameter

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

## 5.16.2.3 void Client.View.Lobby.addLeaveButtonListener ( ActionListener a )

Fuegt einen ActionListener fuer den 'Leave' Button hinzu.

## Parameter

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

5.16.2.4 void Client.View.Lobby.addChatMessageListener ( KeyListener *k* )

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

## Parameter

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

5.16.2.5 void Client.View.Lobby.setLanguage ( Language *l* )

Aendert die Sprache des Fensters.

## Parameter

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

5.16.2.6 void Client.View.Lobby.update ( Observable *o*, Object *arg* )

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

Je nach dem in *arg* uebergebenen ViewNotification-Befehl wird ein Update des Fensters ausgefuehrt 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

5.16.2.7 void Client.View.Lobby.update ( Observable *o*, String *arg* )

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

Dieser wird als Chatnachricht interpretiert und dem Chatlog angefuegt.

## Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<i>arg</i>	erwartet einen String, der eine Chatnachricht darstellt

## 5.17 Client.View.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

#### 5.17.1 Ausführliche Beschreibung

Das Login-Fenster repräsentiert den initialen Dialog zwischen Benutzer und Client. In diesem Fenster kann der Benutzer seinen Namen und die Adresse des Servers eingeben. Ausserdem ist ueber den [Login](#) die Auswahl der Sprache moeglich. ueber den Login-Button wird die Verbindung zum Server hergestellt.

#### 5.17.2 Dokumentation der Elementfunktionen

##### 5.17.2.1 void Client.View.Login.addConnectButtonListener ( ActionListener a )

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

##### Parameter

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

##### 5.17.2.2 void Client.View.Login.addLanguageSelectionListener ( ItemListener i )

Fuegt einen Listener fuer die Sprachauswahl des [Login](#) Fensters hinzu.

##### Parameter

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

##### 5.17.2.3 void Client.View.Login.setLanguage ( Language l )

Aendert die Sprache des Fensters.

##### Parameter

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

##### 5.17.2.4 void Client.View.Login.update ( Observable o, Object arg )

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

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



## Parameter

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

## 5.18 Client.View.OtherPlayer Klassenreferenz

## Private Attribute

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

## 5.18.1 Ausführliche Beschreibung

Zeigt die Informationen über die anderen Spieler an, also den Namen, ein Symbol für die verdeckte Hand und das Label für zusätzliche Angaben.

## 5.19 Client.View.OwnHand Klassenreferenz

## Private Attribute

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

## 5.19.1 Ausführliche Beschreibung

Stellt die Karten dar, die der Spieler auf der Hand hat. Der Spieler kann eine Karte durch Anklicken auswählen und durch einen zweiten Klick ausspielen.

## 5.20 Client.View.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

### 5.20.1 Ausführliche Beschreibung

Dieses Fenster ermöglicht die Eingabe eines Passwortes um einem Passwortgeschuetztem Spiel beizutreten oder per 'Leave' wieder in die [Lobby](#) zurueckzukehren.

### 5.20.2 Dokumentation der Elementfunktionen

#### 5.20.2.1 void Client.View.Password.addJoinButtonListener ( ActionListener a )

Fuegt einen ActionListener fuer den 'Join' Button hinzu.

#### Parameter

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

#### 5.20.2.2 void Client.View.Password.setLanguage ( Language l )

Aendert die Sprache des Fensters.

#### Parameter

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

#### 5.20.2.3 void Client.View.Password.update ( Observable o, Object arg )

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

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

#### Parameter

<i>o</i>	erwartet ein Objekt von der Klasse <a href="#">ClientModel</a>
<i>arg</i>	erwartet: openWarning, passwordAccepted

## 5.21 Client.View.ScoreWindow Klassenreferenz

Abgeleitet von Observer.

### Öffentliche Methoden

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

### 5.21.1 Ausführliche Beschreibung

Dieses Fenster zeigt den momentanen Punktestand nach jeder Runde und den Gesamtpunktestand am Ende des Spieles an.

### 5.21.2 Dokumentation der Elementfunktionen

#### 5.21.2.1 void Client.View.ScoreWindow.update ( Observable o, Object arg )

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

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

Parameter

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

## 5.22 Client.View.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

### 5.22.1 Ausführliche Beschreibung

[ViewCard](#) ist die View-seitige Repraesentation einer Karte. Sie wird verwendet um einzelne Karten auf das Spielfeld zu zeichnen. Dazu enthaelt sie die Pfadangabe zu dem Ordner, in dem die Bilder der Karten gespeichert sind, und eine ID, um das genaue Bild zu spezifizieren.

### 5.22.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.22.2.1 Client.View.ViewCard.ViewCard ( String s, int n )

Erstellt eine neue Karte fuer die Anzeige und zeichnet dafuer das Bild, das durch die Pfadangabe s und seine Kardinaliaet n im Ordner angegeben ist.

Die Pfadangabe wird durch das Regelwerk bestimmt.

## Parameter

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

## 5.22.3 Dokumentation der Elementfunktionen

5.22.3.1 `int Client.View.ViewCard.getID ( )`

Gibt die ID der Karte zurueck.

## Rückgabe

ID der Karte

5.23 `Client.View.Warning` Klassenreferenz

Abgeleitet von `Observer`.

## Öffentliche Methoden

- void `setText` (String text)
- void `update` (Observable o, Object arg)

## Private Attribute

- String **warningText**

## 5.23.1 Ausführliche Beschreibung

Das Warning-Fenster zeigt dem Benutzer Fehlermeldungen bzw. Hinweise an, welche vom `ClientModel` uebergeben wurden. Es wird nur im Fehlerfall angezeigt.

## 5.23.2 Dokumentation der Elementfunktionen

5.23.2.1 `void Client.View.Warning.setText ( String text )`

Setzt den Warnhinweis des Fensters.

## Parameter

<i>text</i>	Warnhinweis, der angezeigt werden soll
-------------	--

5.23.2.2 `void Client.View.Warning.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

## 5.24 Client.ViewNotification Enum-Referenz

### Öffentliche Attribute

- **moveAcknowledged**
- **chooseCardsSuccessful**
- **inputNumberSuccessful**
- **chooseItemSuccessful**
- **playerListUpdate**
- **gameListUpdate**
- **chatMessage**
- **loginSuccessful**
- **joinGameSuccessful**
- **gameStarted**
- **passwordAccepted**
- **playedCardsUpdate**
- **otherDataUpdate**
- **windowChangeForced**
- **openChooseCards**
- **openChooseItem**
- **openInputNumber**
- **openWarning**
- **showScore**

### 5.24.1 Ausführliche Beschreibung

Enum, das vom [ClientModel](#) ueber notify an seine Observer geschickt wird, um mitzuteilen, welche Veraenderung stattgefunden hat.

## 5.25 ComObjects.ComBeenKicked Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

### Öffentliche Methoden

- [ComBeenKicked](#) (String message)
- String [getMessage](#) ()
- void [process](#) ([ClientModel](#) model)
- void [process](#) ([Player](#) player, [Server](#) server)

### Private Attribute

- String **message**

### 5.25.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Die Nachricht wird an einen Spieler gesendet, wenn er aus einem Spiel entfernt wurde. Dies geschieht, wenn ein Spieler ein Spiel verlaesst oder wenn der Spielleiter das Wartefenster verlaesst.

### 5.25.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.25.2.1 ComObjects.ComBeenKicked.ComBeenKicked ( String *message* )

Dies ist der Kontruktor fuer eine neue ComBeenKicked-Nachricht.

Parameter

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

### 5.25.3 Dokumentation der Elementfunktionen

#### 5.25.3.1 String ComObjects.ComBeenKicked.getMessage ( )

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

Rückgabe

die Nachricht.

#### 5.25.3.2 void ComObjects.ComBeenKicked.process ( ClientModel *model* )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

#### 5.25.3.3 void ComObjects.ComBeenKicked.process ( Player *player*, Server *server* )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.26 ComObjects.ComChatMessage Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

Öffentliche Methoden

- [ComChatMessage](#) (String *message*)
- String [getChatMessage](#) ( )
- void [process](#) (ClientModel *model*)
- void [process](#) (Player *player*, Server *server*)

Private Attribute

- String **chatMessage**

## 5.26.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie enthaelt eine Chatnachricht in Form eines Strings.

## 5.26.2 Beschreibung der Konstruktoren und Destruktoren

5.26.2.1 ComObjects.ComChatMessage.ComChatMessage ( String *message* )

Dies ist der Kontruktor fuer eine neue ComChatMessage-Nachricht.

## Parameter

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

## 5.26.3 Dokumentation der Elementfunktionen

## 5.26.3.1 String ComObjects.ComChatMessage.getChatMessage ( )

Hier kann die versendete Nachricht von anderen Klassen ausgelesen werden.

## Rückgabe

die Chatnachricht, die versendet wurde.

5.26.3.2 void ComObjects.ComChatMessage.process ( ClientModel *model* )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

5.26.3.3 void ComObjects.ComChatMessage.process ( Player *player*, Server *server* )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.27 ComObjects.ComClientLeave Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

## Öffentliche Methoden

- [ComClientLeave](#) ()
- void [process](#) (ClientModel *model*)
- void [process](#) (Player *player*, Server *server*)

### 5.27.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie wird zur Benachrichtigung gesendet, wenn ein Spieler ins naechste Fenster moechte und aus dem alten entfernt werden soll.

### 5.27.2 Dokumentation der Elementfunktionen

#### 5.27.2.1 void ComObjects.ComClientLeave.process ( ClientModel model )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

#### 5.27.2.2 void ComObjects.ComClientLeave.process ( Player player, Server server )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.28 ComObjects.ComClientQuit Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

### Öffentliche Methoden

- [ComClientQuit](#) ()
- void [process](#) ([ClientModel](#) model)
- void [process](#) ([Player](#) player, [Server](#) server)

### 5.28.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Die Nachricht wird verschickt, wenn der Spieler ein Fenster schließt.

### 5.28.2 Dokumentation der Elementfunktionen

#### 5.28.2.1 void ComObjects.ComClientQuit.process ( ClientModel model )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.



## Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

5.28.2.2 void ComObjects.ComClientQuit.process ( *Player player*, *Server server* )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.29 ComObjects.ComCreateGameRequest Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

## Öffentliche Methoden

- [ComCreateGameRequest](#) (String name, Enum ruleset, boolean hasPassword, String password)
- String [getGameName](#) ()
- Enum [getRuleset](#) ()
- boolean [hasPassword](#) ()
- String [getPassword](#) ()
- void [process](#) (ClientModel model)
- void [process](#) (Player player, Server server)

## Private Attribute

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

## 5.29.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Diese Nachricht wird versendet, wenn ein neues Spiel erstellt werden soll.

## 5.29.2 Beschreibung der Konstruktoren und Destruktoren

## 5.29.2.1 ComObjects.ComCreateGameRequest.ComCreateGameRequest ( String name, Enum ruleset, boolean hasPassword, String password )

Dies ist der Kontruktor fuer eine neue ComCreateGameRequest-Nachricht.

Wurde kein Passwort gesetzt, bleibt dieses leer.

**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 `ComObjects.ComCreateGameRequest.hasPassword()`.

**5.29.3 Dokumentation der Elementfunktionen****5.29.3.1 String `ComObjects.ComCreateGameRequest.getGameName ( )`**

Diese Methode gibt den Namen des Spiels zurueck.

**Rückgabe**

den Spielnamen.

**5.29.3.2 Enum `ComObjects.ComCreateGameRequest.getRuleset ( )`**

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

**Rückgabe**

das Regelwerk, welches benutzt wird.

**5.29.3.3 boolean `ComObjects.ComCreateGameRequest.hasPassword ( )`**

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

**Rückgabe**

ob es ein Passwort gibt.

Wird benutzt von `ComObjects.ComCreateGameRequest.ComCreateGameRequest()`.

**5.29.3.4 String `ComObjects.ComCreateGameRequest.getPassword ( )`**

Gibt das Passwort zurueck.

Sollte keines gesetzt sein, wird null zurueck gegeben.

**Rückgabe**

das Passwort.

**5.29.3.5 void `ComObjects.ComCreateGameRequest.process ( ClientModel model )`**

Diese Methode ist noetig, damit der `ClientListenerThread` entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

**Parameter**

<i>model</i>	ist das <code>ClientModel</code> , welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

**5.29.3.6 void `ComObjects.ComCreateGameRequest.process ( Player player, Server server )`**

Diese Methode ist noetig, damit der Thread `Player` entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.30 ComObjects.ComInitGameLobby Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

## Öffentliche Methoden

- [ComInitGameLobby](#) (List< String > playerList)
- List< String > [getPlayerList](#) ()
- void [process](#) (ClientModel model)
- void [process](#) (Player player, Server server)

## Private Attribute

- List< String > **playerList**

### 5.30.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie liefert die Liste der Spieler, die sich bereits beim Betreten des Wartefensters darin befinden.

### 5.30.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.30.2.1 ComObjects.ComInitGameLobby.ComInitGameLobby ( List< String > *playerList* )

Dies ist der Kontruktor fuer eine neue ComInitGameLobby-Nachricht.

## Parameter

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

### 5.30.3 Dokumentation der Elementfunktionen

#### 5.30.3.1 List<String> ComObjects.ComInitGameLobby.getPlayerList ( )

Diese Methode liefert die Liste der Player, die sich beim Hinzufuegen eines weiteren Spielers bereits im Wartefenster befinden.

## Rückgabe

die Liste der Spieler, die im Wartefenster sind.

#### 5.30.3.2 void ComObjects.ComInitGameLobby.process ( ClientModel *model* )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

#### 5.30.3.3 void ComObjects.ComInitGameLobby.process ( Player *player*, Server *server* )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

### 5.31 ComObjects.ComInitLobby Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

## Öffentliche Methoden

- [ComInitLobby](#) (List< String > *playerList*, Set *gameList*)
- List< String > [getPlayerList](#) ()
- Set< [GameServerRepresentation](#) > [getGameList](#) ()
- void [process](#) (ClientModel *model*)
- void [process](#) (Player *player*, Server *server*)

## Private Attribute

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

#### 5.31.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie synchronisiert den Client mit der Lobby, wenn er sich mit dem Server verbindet oder nach einem Spiel in die Lobby zurueckkehrt. Dazu enthaelt sie sowohl die *playerList*, als auch die *gameList*.

#### 5.31.2 Beschreibung der Konstruktoren und Destruktoren

##### 5.31.2.1 ComObjects.ComInitLobby.ComInitLobby ( List< String > *playerList*, Set *gameList* )

Dies ist der Kontruktor fuer 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.

## 5.31.3 Dokumentation der Elementfunktionen

## 5.31.3.1 List&lt;String&gt; ComObjects.ComInitLobby.getPlayerList ( )

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

## Rückgabe

die Liste der Spieler.

## 5.31.3.2 Set&lt;GameServerRepresentation&gt; ComObjects.ComInitLobby.getGameList ( )

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

## Rückgabe

die Liste der Spiele.

## 5.31.3.3 void ComObjects.ComInitLobby.process ( ClientModel model )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

## 5.31.3.4 void ComObjects.ComInitLobby.process ( Player player, Server server )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.32 ComObjects.ComJoinRequest Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

## Öffentliche Methoden

- [ComJoinRequest](#) (String [gameMasterName](#), String password)
- String [getGameMasterName](#) ()
- void [process](#) (ClientModel model)
- void [process](#) (Player player, Server server)

## Private Attribute

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

### 5.32.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie ist eine Nachricht, die an den Server gesendet wird, wenn der Spieler einem bestimmten Spiel beitreten will. Dazu enthaelt sie den Namen des Spielleiters als String und ein Passwort, falls dieses von Spielleiter gesetzt wurde.

### 5.32.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.32.2.1 ComObjects.ComJoinRequest.ComJoinRequest ( String *gameMasterName*, String *password* )

Dies ist der Kontruktor fuer 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>password</i>	fuer das Spiel.

Benutzt ComObjects.ComJoinRequest.gameMasterName.

### 5.32.3 Dokumentation der Elementfunktionen

#### 5.32.3.1 String ComObjects.ComJoinRequest.getGameMasterName ( )

Diese Methode gibt den Namen des Spielleiters zurueck.

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

Rückgabe

der Name des Spielleiters.

Benutzt ComObjects.ComJoinRequest.gameMasterName.

#### 5.32.3.2 void ComObjects.ComJoinRequest.process ( ClientModel *model* )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

#### 5.32.3.3 void ComObjects.ComJoinRequest.process ( Player *player*, Server *server* )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

### 5.32.4 Dokumentation der Datenelemente

#### 5.32.4.1 String ComObjects.ComJoinRequest.gameMasterName [private]

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

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

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

### 5.33 ComObjects.ComKickPlayerRequest Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

#### Öffentliche Methoden

- [ComKickPlayerRequest](#) (String [playerName](#))
- String [getPlayerName](#) ()
- void [process](#) (ClientModel model)
- void [process](#) (Player player, [Server](#) server)

#### Private Attribute

- String [playerName](#)

#### 5.33.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie ist eine Nachricht an den Server, die angibt einen Spieler vom Spiel zu entfernen. Dazu enthaelt sie einen String, der den Namen des Spielers enthaelt.

#### 5.33.2 Beschreibung der Konstruktoren und Destruktoren

##### 5.33.2.1 ComObjects.ComKickPlayerRequest.ComKickPlayerRequest ( String *playerName* )

Dies ist der Kontruktor fuer eine neue ComKickPlayerRequest-Nachricht.

Diese enthaelt den Namen des Spielers, der aus den Spiel geloescht werden soll.

#### Parameter

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

Benutzt ComObjects.ComKickPlayerRequest.playerName.

#### 5.33.3 Dokumentation der Elementfunktionen

##### 5.33.3.1 String ComObjects.ComKickPlayerRequest.getPlayerName ( )

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

#### Rückgabe

den Spielernamen.

Benutzt ComObjects.ComKickPlayerRequest.playerName.

#### 5.33.3.2 void ComObjects.ComKickPlayerRequest.process ( ClientModel *model* )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.



## Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

5.33.3.3 void ComObjects.ComKickPlayerRequest.process ( Player *player*, Server *server* )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.34 ComObjects.ComLobbyUpdateGamelist Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

## Öffentliche Methoden

- [ComLobbyUpdateGamelist](#) (boolean removeFlag, [GameServerRepresentation](#) gameServer)
- boolean [isRemoveFlag](#) ()
- [GameServerRepresentation](#) [getGameServer](#) ()
- void [process](#) (ClientModel model)
- void [process](#) (Player player, Server server)

## Private Attribute

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

## 5.34.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie aktualisiert die Gameliste in der Lobby. Dazu enthaelt sie den GameServer und ein RemoveFlag.

## 5.34.2 Beschreibung der Konstruktoren und Destruktoren

5.34.2.1 ComObjects.ComLobbyUpdateGamelist.ComLobbyUpdateGamelist ( boolean *removeFlag*, [GameServerRepresentation](#) *gameServer* )

Dies ist der Kontruktor fuer eine neue ComLobbyUpdateGamelist-Nachricht.

## Parameter

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

### 5.34.3 Dokumentation der Elementfunktionen

#### 5.34.3.1 `boolean ComObjects.ComLobbyUpdateGamelist.isRemoveFlag ( )`

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

##### Rückgabe

ob das Spiel gelöscht wird.

#### 5.34.3.2 `GameServerRepresentation ComObjects.ComLobbyUpdateGamelist.getServer ( )`

Diese Methode liefert das Spiel, das geupdated werden soll.

##### Rückgabe

das Spiel.

#### 5.34.3.3 `void ComObjects.ComLobbyUpdateGamelist.process ( ClientModel model )`

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

#### 5.34.3.4 `void ComObjects.ComLobbyUpdateGamelist.process ( Player player, Server server )`

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.35 ComObjects.ComLoginRequest Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

### Öffentliche Methoden

- [ComLoginRequest](#) (String name)
- String [getPlayerName](#) ()
- void [process](#) (ClientModel model)
- void [process](#) (Player player, Server server)

### Private Attribute

- String **playerName**

### 5.35.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie ist eine Nachricht, die beim Login an den Server gesendet wird. Dazu enthaelt sie den Namen des Spielers, der sich einloggen moechte.

### 5.35.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.35.2.1 ComObjects.ComLoginRequest.ComLoginRequest ( String *name* )

Dies ist der Kontruktor fuer eine neue ComLoginRequest-Nachricht.

##### Parameter

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

### 5.35.3 Dokumentation der Elementfunktionen

#### 5.35.3.1 String ComObjects.ComLoginRequest.getPlayerName ( )

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

Dieser muss auf Eindeutigkeit geprueft werden.

##### Rückgabe

den Spielernamen.

#### 5.35.3.2 void ComObjects.ComLoginRequest.process ( ClientModel *model* )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

#### 5.35.3.3 void ComObjects.ComLoginRequest.process ( Player *player*, Server *server* )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.36 ComObjects.ComObject Schnittstellenreferenz

Basisklasse für [ComObjects.ComBeenKicked](#), [ComObjects.ComChatMessage](#), [ComObjects.ComClientLeave](#), [ComObjects.ComClientQuit](#), [ComObjects.ComCreateGameRequest](#), [ComObjects.ComInitGameLobby](#), [ComObjects.ComInitLobby](#), [ComObjects.ComJoinRequest](#), [ComObjects.ComKickPlayerRequest](#), [ComObjects.ComLobbyUpdateGamelist](#), [ComObjects.ComLoginRequest](#), [ComObjects.ComRuleset](#), [ComObjects.ComServerAcknowledgement](#), [ComObjects.ComStartGame](#), [ComObjects.ComUpdatePlayerlist](#) und [ComObjects.ComWarning](#).

## Öffentliche Methoden

- void [process](#) ([ClientModel](#) model)
- void [process](#) ([Player](#) player, [Server](#) server)

### 5.36.1 Ausführliche Beschreibung

Die Klasse [ComObject](#) ist ein Interface, welches ein Objekt darstellt, das zur Kommunikation genutzt werden kann. Spezielle [ComObject](#) Klassen implementieren diese grundlegenden Klasse.

### 5.36.2 Dokumentation der Elementfunktionen

#### 5.36.2.1 void ComObjects.ComObject.process ( [ClientModel](#) model )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert in [ComObjects.ComCreateGameRequest](#), [ComObjects.ComInitLobby](#), [ComObjects.ComJoinRequest](#), [ComObjects.ComUpdatePlayerlist](#), [ComObjects.ComLobbyUpdateGamelist](#), [ComObjects.ComKickPlayerRequest](#), [ComObjects.ComInitGameLobby](#), [ComObjects.ComBeenKicked](#), [ComObjects.ComRuleset](#), [ComObjects.ComWarning](#), [ComObjects.ComLoginRequest](#), [ComObjects.ComChatMessage](#), [ComObjects.ComClientLeave](#), [ComObjects.ComClientQuit](#), [ComObjects.ComStartGame](#) und [ComObjects.ComServerAcknowledgement](#).

#### 5.36.2.2 void ComObjects.ComObject.process ( [Player](#) player, [Server](#) server )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert in [ComObjects.ComCreateGameRequest](#), [ComObjects.ComInitLobby](#), [ComObjects.ComJoinRequest](#), [ComObjects.ComUpdatePlayerlist](#), [ComObjects.ComLobbyUpdateGamelist](#), [ComObjects.ComKickPlayerRequest](#), [ComObjects.ComInitGameLobby](#), [ComObjects.ComBeenKicked](#), [ComObjects.ComRuleset](#), [ComObjects.ComWarning](#), [ComObjects.ComLoginRequest](#), [ComObjects.ComChatMessage](#), [ComObjects.ComClientLeave](#), [ComObjects.ComClientQuit](#), [ComObjects.ComStartGame](#) und [ComObjects.ComServerAcknowledgement](#).

## 5.37 ComObjects.ComRuleset Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

## Öffentliche Methoden

- [ComRuleset](#) ([RulesetMessage](#) rulesetMessage)
- [RulesetMessage](#) getRulesetMessage ()
- void [process](#) ([ClientModel](#) model)
- void [process](#) ([Player](#) player, [Server](#) server)

## Private Attribute

- [RulesetMessage](#) rulesetMessage

## 5.37.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie ist die grundlegende Nachricht eines Regelwerkaufrufes und enthaelt eine verfeinerte Nachricht mit weiteren Informationen, die [RulesetMessage](#).

## 5.37.2 Beschreibung der Konstruktoren und Destruktoren

5.37.2.1 ComObjects.ComRuleset.ComRuleset ( [RulesetMessage rulesetMessage](#) )

Dies ist der Kontruktor fuer eine neue ComRuleset-Nachricht.

Parameter

<a href="#">rulesetMessage</a>	ist eine Nachricht, die ans Ruleset gesendet werden soll.
--------------------------------	---

## 5.37.3 Dokumentation der Elementfunktionen

5.37.3.1 [RulesetMessage](#) ComObjects.ComRuleset.getRulesetMessage ( )

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

Rückgabe

die Nachricht.

5.37.3.2 void ComObjects.ComRuleset.process ( [ClientModel model](#) )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

5.37.3.3 void ComObjects.ComRuleset.process ( [Player player](#), [Server server](#) )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.38 ComObjects.ComServerAcknowledgement Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

Öffentliche Methoden

- void [process](#) ([ClientModel](#) model)
- void [process](#) ([Player](#) player, [Server](#) server)

### 5.38.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Diese Nachricht wird vom Server als Bestätigung gesendet.

### 5.38.2 Dokumentation der Elementfunktionen

#### 5.38.2.1 void ComObjects.ComServerAcknowledgement.process ( ClientModel model )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

#### 5.38.2.2 void ComObjects.ComServerAcknowledgement.process ( Player player, Server server )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.39 ComObjects.ComStartGame Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

### Öffentliche Methoden

- [ComStartGame](#) ()
- void [process](#) ([ClientModel](#) model)
- void [process](#) ([Player](#) player, [Server](#) server)

### 5.39.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie wird versendet, wenn ein Spiel gestartet werden soll.

### 5.39.2 Dokumentation der Elementfunktionen

#### 5.39.2.1 void ComObjects.ComStartGame.process ( ClientModel model )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

##### Parameter

---

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

#### 5.39.2.2 void ComObjects.ComStartGame.process ( Player player, Server server )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.40 ComObjects.ComUpdatePlayerlist Klassenreferenz

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

Öffentliche Methoden

- [ComUpdatePlayerlist](#) (String playerName, boolean removeFlag)
- String [getPlayerName](#) ()
- boolean [isRemoveFlag](#) ()
- void [process](#) (ClientModel model)
- void [process](#) (Player player, Server server)

Private Attribute

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

### 5.40.1 Ausführliche Beschreibung

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie sendet eine Nachricht zum Update der Playerliste in der Lobby und Spiellobby. Dazu enthaelt sie den Player und ein removeFlag.

### 5.40.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.40.2.1 ComObjects.ComUpdatePlayerlist.ComUpdatePlayerlist ( String playerName, boolean removeFlag )

Dies ist der Kontruktor fuer eine neue ComUpdatePlayerlist-Nachricht.

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

Parameter

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

### 5.40.3 Dokumentation der Elementfunktionen

#### 5.40.3.1 String ComObjects.ComUpdatePlayerlist.getPlayerName ( )

Diese Methode gibt den Namen des Spielers zurueck.

**Rückgabe**

den Spielernamen.

**5.40.3.2 boolean ComObjects.ComUpdatePlayerlist.isRemoveFlag ( )**

Diese Methode gibt zurueck, ob der Spieler aus der Liste geloescht werden soll oder nicht.

**Rückgabe**

ob der Spieler geloescht werden soll.

**5.40.3.3 void ComObjects.ComUpdatePlayerlist.process ( ClientModel model )**

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

**Parameter**

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

**5.40.3.4 void ComObjects.ComUpdatePlayerlist.process ( Player player, Server server )**

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

**Parameter**

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

**5.41 ComObjects.ComWarning Klassenreferenz**

Abgeleitet von [ComObjects.ComObject](#) und Serializable.

**Öffentliche Methoden**

- [ComWarning](#) (String warning)
- String [getWarning](#) ()
- void [process](#) (ClientModel model)
- void [process](#) (Player player, Server server)

**Private Attribute**

- String **warning**

**5.41.1 Ausführliche Beschreibung**

Diese Klasse ist ein spezielles Kommunikations-Objekt. Sie soll dem Spieler eine Mitteilung senden und so ueber ein Fehlerevent informieren.



## 5.41.2 Beschreibung der Konstruktoren und Destruktoren

5.41.2.1 ComObjects.ComWarning.ComWarning ( String *warning* )

Dies ist der Konstruktor einer neuen ComWarning-Nachricht.

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

Parameter

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

## 5.41.3 Dokumentation der Elementfunktionen

## 5.41.3.1 String ComObjects.ComWarning.getWarning ( )

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

Rückgabe

die Warnnachricht.

5.41.3.2 void ComObjects.ComWarning.process ( ClientModel *model* )

Diese Methode ist noetig, damit der ClientListenerThread entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>model</i>	ist das ClientModel, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
--------------	--

Implementiert [ComObjects.ComObject](#).

5.41.3.3 void ComObjects.ComWarning.process ( Player *player*, Server *server* )

Diese Methode ist noetig, damit der Thread Player entscheiden kann welche Message das Object enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>player</i>	Der Client welcher den Aufruf startet.
<i>server</i>	Der Server an den sich das ComObjekt weitergibt.

Implementiert [ComObjects.ComObject](#).

## 5.42 ComObjects.MsgCard Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und Serializable.

Öffentliche Methoden

- [MsgCard](#) (Card card)
- [Card](#) getCard ()
- void visit (ServerRuleset serverRuleset, String name)
- void visit (ClientRuleset clientRuleset)

Private Attribute

- [Card](#) card

### 5.42.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Sie beinhaltet die ausgespielte Karte eines Spielers.

### 5.42.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.42.2.1 ComObjects.MsgCard.MsgCard ( Card card )

Dies ist der Kontruktor fuer eine neue MsgCard-Nachricht.

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

Parameter

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

### 5.42.3 Dokumentation der Elementfunktionen

#### 5.42.3.1 Card ComObjects.MsgCard.getCard ( )

Diese Methode gibt die ausgespielte Karte des Spielers zurueck.

Rückgabe

die Karte.

#### 5.42.3.2 void ComObjects.MsgCard.visit ( ServerRuleset serverRuleset, String name )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

#### 5.42.3.3 void ComObjects.MsgCard.visit ( ClientRuleset clientRuleset )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

## 5.43 ComObjects.MsgCardRequest Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und Serializable.

Öffentliche Methoden

- [MsgCardRequest](#) ( )

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

#### 5.43.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Diese Nachricht wird von Server gesendet, um einem Spieler mitzuteilen, dass er das Spielen einer Karte erwartet.

#### 5.43.2 Dokumentation der Elementfunktionen

##### 5.43.2.1 void ComObjects.MsgCardRequest.visit ( [ServerRuleset](#) *serverRuleset*, *String name* )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

##### 5.43.2.2 void ComObjects.MsgCardRequest.visit ( [ClientRuleset](#) *clientRuleset* )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

## 5.44 ComObjects.MsgGameEnd Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und Serializable.

#### Öffentliche Methoden

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

#### Private Attribute

- String **winnerName**

#### 5.44.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Sie signalisiert dem ClientRuleset, dass das Spiel zu Ende ist.

#### 5.44.2 Beschreibung der Konstruktoren und Destruktoren

##### 5.44.2.1 ComObjects.MsgGameEnd.MsgGameEnd ( String *name* )

Dies ist der Kontruktor fuer eine neue MsgGameEnd-Nachricht.

## Parameter

<i>name</i>	ist der Name des Gewinners.
-------------	-----------------------------

## 5.44.3 Dokumentation der Elementfunktionen

## 5.44.3.1 String ComObjects.MsgGameEnd.getWinnerName ( )

Diese Methode liefert den Namen des Gewinners eines Spiels.

## Rückgabe

den Gewinnernamen.

## 5.44.3.2 void ComObjects.MsgGameEnd.visit ( ServerRuleset serverRuleset, String name )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

## 5.44.3.3 void ComObjects.MsgGameEnd.visit ( ClientRuleset clientRuleset )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

## 5.45 ComObjects.MsgMultiCards Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und Serializable.

## Öffentliche Methoden

- [MsgMultiCards](#) (Set cardList)
- Set< [Card](#) > [getCardList](#) ()
- void [visit](#) ([ServerRuleset](#) serverRuleset, String name)
- void [visit](#) ([ClientRuleset](#) clientRuleset)

## Private Attribute

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

## 5.45.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Sie liefert mehrere Karten zum Tausch fuer das Regelwerk Hearts.

### 5.45.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.45.2.1 ComObjects.MsgMultiCards.MsgMultiCards ( Set *cardList* )

Dies ist der Kontruktor fuer eine neue MsgMultiCards-Nachricht.

Parameter

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

### 5.45.3 Dokumentation der Elementfunktionen

#### 5.45.3.1 Set<Card> ComObjects.MsgMultiCards.getCardList ( )

Gibt die Liste der gewaehlten Karten zurueck.

Rückgabe

die Liste der Karten.

#### 5.45.3.2 void ComObjects.MsgMultiCards.visit ( ServerRuleset *serverRuleset*, String *name* )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

#### 5.45.3.3 void ComObjects.MsgMultiCards.visit ( ClientRuleset *clientRuleset* )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

## 5.46 ComObjects.MsgMultiCardsRequest Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und Serializable.

Öffentliche Methoden

- [MsgMultiCardsRequest](#) (int *count*)
- int [getCount](#) ()
- void [visit](#) (ServerRuleset *serverRuleset*, String *name*)
- void [visit](#) (ClientRuleset *clientRuleset*)

Private Attribute

- int [count](#)

## 5.46.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Diese Nachricht wird gesendet, wenn die Auswahl mehrerer Karten vom Spieler gefordert werden soll.

## 5.46.2 Beschreibung der Konstruktoren und Destruktoren

## 5.46.2.1 ComObjects.MsgMultiCardsRequest.MsgMultiCardsRequest ( int count )

Dies ist der Kontruktor fuer eine neue MsgMultipleCardsRequest-Nachricht.

## Parameter

<i>count</i>	ist die erwartete Anzahl an Karten.
--------------	-------------------------------------

Benutzt ComObjects.MsgMultiCardsRequest.count.

## 5.46.3 Dokumentation der Elementfunktionen

## 5.46.3.1 int ComObjects.MsgMultiCardsRequest.getCount ( )

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

## Rückgabe

die Anzahl der Karten.

Benutzt ComObjects.MsgMultiCardsRequest.count.

## 5.46.3.2 void ComObjects.MsgMultiCardsRequest.visit ( ServerRuleset serverRuleset, String name )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

## 5.46.3.3 void ComObjects.MsgMultiCardsRequest.visit ( ClientRuleset clientRuleset )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

## Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

## 5.47 ComObjects.MsgNumber Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und Serializable.

## Öffentliche Methoden

- [MsgNumber](#) (int number)

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

#### Private Attribute

- int **number**

#### 5.47.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Sie enthaelt eine Zahl, die ein Spieler zuvor ausgewaehlt hat.

#### 5.47.2 Beschreibung der Konstruktoren und Destruktoren

##### 5.47.2.1 ComObjects.MsgNumber.MsgNumber ( int *number* )

Dies ist der Kontruktor fuer eine neue MsgNumber-Nachricht.

#### Parameter

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

#### 5.47.3 Dokumentation der Elementfunktionen

##### 5.47.3.1 int ComObjects.MsgNumber.getNumber ( )

Diese Methode liefert die Eingabe eines Spielers.

#### Rückgabe

eine Zahl, die Eingabe des Spielers.

##### 5.47.3.2 void ComObjects.MsgNumber.visit ( [ServerRuleset](#) *serverRuleset*, String *name* )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

#### Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

##### 5.47.3.3 void ComObjects.MsgNumber.visit ( [ClientRuleset](#) *clientRuleset* )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

#### Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).



## 5.48 ComObjects.MsgNumberRequest Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und [Serializable](#).

### Öffentliche Methoden

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

### 5.48.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Sie Wird gesendet, wenn die Eingabe einer Zahl gefordert werden soll.

### 5.48.2 Dokumentation der Elementfunktionen

#### 5.48.2.1 void ComObjects.MsgNumberRequest.visit ( [ServerRuleset](#) serverRuleset, String name )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

#### Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

#### 5.48.2.2 void ComObjects.MsgNumberRequest.visit ( [ClientRuleset](#) clientRuleset )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

#### Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

## 5.49 ComObjects.MsgSelection Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und [Serializable](#).

### Öffentliche Methoden

- [MsgSelection](#) ([Colour](#) selection)
- [Colour](#) getSelection ()
- void [visit](#) ([ServerRuleset](#) serverRuleset, String name)
- void [visit](#) ([ClientRuleset](#) clientRuleset)

### Private Attribute

- [Colour](#) selection

### 5.49.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Diese Nachricht enthaelt eine Kartenfarbe, die der Spieler zuvor ausgewaehlt hat.

### 5.49.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.49.2.1 ComObjects.MsgSelection.MsgSelection ( Colour selection )

Dies ist der Kontruktor fuer eine neue MsgSelection-Nachricht.

##### Parameter

<i>selection</i>	ist die Farbe der Karte, die der Spieler gewaehlt hat.
------------------	--

### 5.49.3 Dokumentation der Elementfunktionen

#### 5.49.3.1 Colour ComObjects.MsgSelection.getSelection ( )

Diese Methode gibt die Farbe zurueck, die der Spieler gewaehlt hat.

##### Rückgabe

die gewaehlte Farbe.

#### 5.49.3.2 void ComObjects.MsgSelection.visit ( ServerRuleset serverRuleset, String name )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

#### 5.49.3.3 void ComObjects.MsgSelection.visit ( ClientRuleset clientRuleset )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

## 5.50 ComObjects.MsgSelectionRequest Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und Serializable.

### Öffentliche Methoden

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

### 5.50.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Diese Nachricht sendet der Server an einen Spieler, wenn er eine Farbauswahl von diesem erwartet.

### 5.50.2 Dokumentation der Elementfunktionen

#### 5.50.2.1 void ComObjects.MsgSelectionRequest.visit ( ServerRuleset serverRuleset, String name )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

#### 5.50.2.2 void ComObjects.MsgSelectionRequest.visit ( ClientRuleset clientRuleset )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

##### Parameter

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

## 5.51 ComObjects.MsgUser Klassenreferenz

Abgeleitet von [ComObjects.RulesetMessage](#) und Serializable.

### Öffentliche Methoden

- [MsgUser](#) ([GameClientUpdate](#) gameClientUpdate)
- [GameClientUpdate](#) [getGameClientUpdate](#) ()
- void [visit](#) ([ServerRuleset](#) serverRuleset, String name)
- void [visit](#) ([ClientRuleset](#) clientRuleset)

### Private Attribute

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

### 5.51.1 Ausführliche Beschreibung

Diese Klasse ist eine Verfeinerung der RulesetMessage-Klasse. Sie wird dem Client gesendet, um dem Client-Ruleset den aktuellen Spielzustand in Form eines GameClientUpdate zu uebermitteln.

### 5.51.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.51.2.1 ComObjects.MsgUser.MsgUser ( GameClientUpdate gameClientUpdate )

Dies ist der Konstruktor einer neuen MsgUser-Nachricht.

**Parameter**

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

**5.51.3 Dokumentation der Elementfunktionen****5.51.3.1 GameClientUpdate ComObjects.MsgUser.getGameClientUpdate ( )**

Diese Methode liefert den den aktuellen Spielzustand, der fuer ein Update benoetigt wird.

**Rückgabe**

den aktuellen Spielzustand.

**5.51.3.2 void ComObjects.MsgUser.visit ( ServerRuleset serverRuleset, String name )**

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

**Parameter**

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert [ComObjects.RulesetMessage](#).

**5.51.3.3 void ComObjects.MsgUser.visit ( ClientRuleset clientRuleset )**

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

**Parameter**

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert [ComObjects.RulesetMessage](#).

**5.52 ComObjects.RulesetMessage Schnittstellenreferenz**

Basisklasse für [ComObjects.MsgCard](#), [ComObjects.MsgCardRequest](#), [ComObjects.MsgGameEnd](#), [ComObjects.-MsgMultiCards](#), [ComObjects.MsgMultiCardsRequest](#), [ComObjects.MsgNumber](#), [ComObjects.MsgNumber-Request](#), [ComObjects.MsgSelection](#), [ComObjects.MsgSelectionRequest](#) und [ComObjects.MsgUser](#).

**Öffentliche Methoden**

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

**5.52.1 Ausführliche Beschreibung**

Dieses Interface ist eine Verfeinerung der ComRuleset-Klasse. Es enthaelt Methoden, die von speziellen Ruleset-Messages implementiert werden müssen.

### 5.52.2 Dokumentation der Elementfunktionen

#### 5.52.2.1 void ComObjects.RulesetMessage.visit ( *ServerRuleset serverRuleset*, String *name* )

Diese Methode ist noetig, damit das ServerRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

**Parameter**

<i>serverRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
<i>name</i>	ist der Name des Spielers.

Implementiert in [ComObjects.MsgMultiCardsRequest](#), [ComObjects.MsgSelection](#), [ComObjects.MsgCard](#), [ComObjects.MsgUser](#), [ComObjects.MsgGameEnd](#), [ComObjects.MsgMultiCards](#), [ComObjects.MsgNumber](#), [ComObjects.MsgCardRequest](#), [ComObjects.MsgNumberRequest](#) und [ComObjects.MsgSelectionRequest](#).

#### 5.52.2.2 void ComObjects.RulesetMessage.visit ( ClientRuleset clientRuleset )

Diese Methode ist noetig, damit das ClientRuleset entscheiden kann welche Message es enthaelt und wie diese verarbeitet werden soll.

**Parameter**

<i>clientRuleset</i>	ist das Ruleset, welches uebergeben wird, damit die ueberladene Methode richtig gewaehlt wird.
----------------------	--

Implementiert in [ComObjects.MsgMultiCardsRequest](#), [ComObjects.MsgCard](#), [ComObjects.MsgSelection](#), [ComObjects.MsgUser](#), [ComObjects.MsgGameEnd](#), [ComObjects.MsgMultiCards](#), [ComObjects.MsgNumber](#), [ComObjects.MsgNumberRequest](#), [ComObjects.MsgSelectionRequest](#) und [ComObjects.MsgCardRequest](#).

### 5.53 Ruleset.Card Schnittstellenreferenz

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

**Öffentliche Methoden**

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

#### 5.53.1 Ausführliche Beschreibung

Dieses Interface modelliert eine Spielkarte

#### 5.53.2 Dokumentation der Elementfunktionen

##### 5.53.2.1 int Ruleset.Card.getValue ( )

Gibt den Wert der Karte zurück.

**Rückgabe**

Der Wert der Karte

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

##### 5.53.2.2 Colour Ruleset.Card.getColour ( )

Gibt die Farbe der Karte zurück.

**Rückgabe**

Die Farbe der Karte

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

## 5.54 Ruleset.ClientHearts Klassenreferenz

Abgeleitet von [Ruleset.ClientRuleset](#).

### Öffentliche Methoden

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

### Statische, private Attribute

- static final int [MIN\\_PLAYERS](#) = 4
- static final int [MAX\\_PLAYERS](#) = 4
- static final [RulesetType](#) RULESET = [RulesetType.Hearts](#)

### Weitere Geerbte Elemente

#### 5.54.1 Ausführliche Beschreibung

Diese Klasse bildet das Regelwerk für den Clientmodel bei einer Partie Hearts

#### 5.54.2 Beschreibung der Konstruktoren und Destruktoren

##### 5.54.2.1 Ruleset.ClientHearts.ClientHearts ( [ClientModel](#) client )

Erzeugt ein [ClientHearts](#).

#### Parameter

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

Benutzt [Ruleset.RulesetType.Hearts](#), [Ruleset.ClientHearts.MAX\\_PLAYERS](#) und [Ruleset.ClientHearts.MIN\\_PLAYERS](#).

#### 5.54.3 Dokumentation der Elementfunktionen

##### 5.54.3.1 boolean Ruleset.ClientHearts.isValidMove ( [Card](#) card ) [virtual]

Prueft ob ein gemachter Zug in einem Spiel gueltig war.

#### Parameter

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

#### Rückgabe

true falls die Karte gueltig ist, false wenn nicht

Implementiert [Ruleset.ClientRuleset](#).

##### 5.54.3.2 void Ruleset.ClientHearts.resolveMessage ( [MsgMultiCardsRequest](#) msgMultiCardsRequest )

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

## Parameter

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

## 5.54.3.3 boolean Ruleset.ClientHearts.isValidChosenCards ( Set&lt; Card &gt; cards )

Gibt zuueck ob die Karten die der Client tauschen will, gueltig sind.

## Parameter

<i>cards</i>	Die zu tauschenden Karten
--------------	---------------------------

## Rückgabe

true wenn Karten valide sind, false wenn nicht

## 5.55 Ruleset.ClientRuleset Klassenreferenz

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

## Öffentliche Methoden

- [ClientRuleset](#) ([RulesetType](#) ruleset, int minPlayers, int maxPlayers, [ClientModel](#) client)
- [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 resolveMessage ([MsgGameEnd](#) msgCardRequest)
- abstract boolean isValidMove ([Card](#) card)

## Geschützte Methoden

- void announceWinner (String winner)
- void send ([RulesetMessage](#) message)

## Private Attribute

- [ClientModel](#) client
- [GameClientUpdate](#) gameState
- final [RulesetType](#) RULESET
- final int MIN\_PLAYERS
- final int MAX\_PLAYERS
- [GamePhase](#) gamePhase



### 5.55.1 Ausführliche Beschreibung

[ClientRuleset](#) ist eine abstrakte Klasse und wird zur Regelvorauswertung im Client verwendet. Dazu benutzt es die [isValidMove\(\)](#) Methode. Des Weiteren kann es vom ClientModel erhaltene RulesetMessages mit der [resolveMessage\(\)](#) Methode behandeln und neue RulesetMessages senden.

### 5.55.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.55.2.1 Ruleset.ClientRuleset.ClientRuleset ( RulesetType ruleset, int minPlayers, int maxPlayers, ClientModel client )

Erstellt eine [ClientRuleset](#) Klasse.

Parameter

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

Benutzt Ruleset.ClientRuleset.client, Ruleset.ClientRuleset.gamePhase, Ruleset.ClientRuleset.MAX\_PLAYERS, Ruleset.ClientRuleset.MIN\_PLAYERS, Ruleset.ClientRuleset.RULESET und Ruleset.GamePhase.Start.

### 5.55.3 Dokumentation der Elementfunktionen

#### 5.55.3.1 RulesetType Ruleset.ClientRuleset.getRulesetType ( )

Gibt den Typ des Regelwerks zurueck.

Rückgabe

Der Typ vom Regelwerk

Benutzt Ruleset.ClientRuleset.RULESET.

#### 5.55.3.2 int Ruleset.ClientRuleset.getMinPlayers ( )

Gibt die Mindestanzahl an Spielern zurueck für dieses Spiel.

Rückgabe

Die Mindestanzahl an Spielern

Benutzt Ruleset.ClientRuleset.MIN\_PLAYERS.

#### 5.55.3.3 int Ruleset.ClientRuleset.getMaxPlayers ( )

Gibt die Maximale Anzahl an Spielern zurueck.

Rückgabe

Die maximale Anzahl an Spielern

Benutzt Ruleset.ClientRuleset.MAX\_PLAYERS.

#### 5.55.3.4 GamePhase Ruleset.ClientRuleset.getGamePhase ( )

Gibt die momentane Spielphase zurück.

**Rückgabe**

gamePhase Die Spielphase

Benutzt Ruleset.ClientRuleset.gamePhase.

**5.55.3.5 List<Card> Ruleset.ClientRuleset.getOwnHand ( )**

Gibt die eigenen Handkarten zurueck.

**Rückgabe**

Liste von Karten

**5.55.3.6 OtherData Ruleset.ClientRuleset.getOwnData ( )**

Gibt die [OtherData](#) des Models zurueck.

**Rückgabe**

Die Otherdata des Models

**5.55.3.7 OtherData Ruleset.ClientRuleset.getOtherPlayerData ( String player )**

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

**Parameter**

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

**Rückgabe**

otherPlayerData Die [OtherData](#)

**5.55.3.8 PlayerState Ruleset.ClientRuleset.getCurrentPlayer ( )**

Gibt den Spieler der momentan am Zug ist zurueck.

**Rückgabe**

Der momentane Spieler

**5.55.3.9 Card Ruleset.ClientRuleset.getTrumpCard ( )**

Holt die aufgedeckte Trumpfkarte.

**Rückgabe**

Eine Karte

**5.55.3.10 void Ruleset.ClientRuleset.resolveMessage ( RulesetMessage message )**

Verarbeitet eine RulesetMessage vom Server.

## Parameter

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

5.55.3.11 void Ruleset.ClientRuleset.resolveMessage ( MsgUser *clientUpdate* )

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

## Parameter

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

5.55.3.12 void Ruleset.ClientRuleset.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
------------------------	--------------------------

5.55.3.13 void Ruleset.ClientRuleset.resolveMessage ( MsgGameEnd *msgCardRequest* )

Verarbeitet die RulesetMessage dass das Spiel zu Ende ist und das es einen Sieger gibt.

## Parameter

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

5.55.3.14 void Ruleset.ClientRuleset.announceWinner ( String *winner* ) [protected]

Ruft beim Model die Methode announceWinner, wenn es einem Gewinner gibt.

## Parameter

<i>winner</i>	Der Gewinner
---------------	--------------

5.55.3.15 void Ruleset.ClientRuleset.send ( RulesetMessage *message* ) [protected]

Ruft beim Model die send Methode auf und verschickt eine Rulesetmessage.

## Parameter

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

5.55.3.16 abstract boolean Ruleset.ClientRuleset.isValidMove ( Card *card* ) [pure virtual]

Prueft ob ein gemachter Zug in einem Spiel gueltig war.

## Parameter

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

## Rückgabe

true falls die Karte gueltig ist, false wenn nicht

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

## 5.56 Ruleset.ClientWizard Klassenreferenz

Abgeleitet von [Ruleset.ClientRuleset](#).

### Öffentliche Methoden

- [ClientWizard](#) ([ClientModel](#) client)
- boolean [isValidMove](#) ([Card](#) card)
- void [resolveMessage](#) ([MsgNumberRequest](#) msgNumber)
- void [resolveMessage](#) ([MsgSelectionRequest](#) msgSelection)
- boolean [isValidTrickNumber](#) (int number)
- boolean [isValidColour](#) ([Colour](#) colour)

### Statische, private Attribute

- static final int [MIN\\_PLAYERS](#) = 3
- static final int [MAX\\_PLAYERS](#) = 6
- static final [RulesetType](#) RULESET = [RulesetType.Wizard](#)

### Weitere Geerbte Elemente

#### 5.56.1 Ausführliche Beschreibung

Diese Klasse bildet das Regelwerk fuer den Client bei einer Partie Wizard

#### 5.56.2 Beschreibung der Konstruktoren und Destruktoren

##### 5.56.2.1 Ruleset.ClientWizard.ClientWizard ( [ClientModel](#) client )

Erzeugt ein [ClientWizard](#).

#### Parameter

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

Benutzt [Ruleset.ClientWizard.MAX\\_PLAYERS](#), [Ruleset.ClientWizard.MIN\\_PLAYERS](#) und [Ruleset.ClientWizard.RULESET](#).

#### 5.56.3 Dokumentation der Elementfunktionen

##### 5.56.3.1 boolean Ruleset.ClientWizard.isValidMove ( [Card](#) card ) [virtual]

Prueft ob ein gemachter Zug in einem Spiel gueltig war.

#### Parameter

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

#### Rückgabe

true falls die Karte gueltig ist, false wenn nicht

Implementiert [Ruleset.ClientRuleset](#).

##### 5.56.3.2 void Ruleset.ClientWizard.resolveMessage ( [MsgNumberRequest](#) msgNumber )

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

## Parameter

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

5.56.3.3 void Ruleset.ClientWizard.resolveMessage ( MsgSelectionRequest *msgSelection* )

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

## Parameter

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

5.56.3.4 boolean Ruleset.ClientWizard.isValidTrickNumber ( int *number* )

Prüft ob die Anzahl der angesagten Stiche vom Spieler gueltig sind.

## Parameter

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

## Rückgabe

true falls die Anzahl der Stiche passen, false wenn nicht

5.56.3.5 boolean Ruleset.ClientWizard.isValidColour ( Colour *colour* )

Prüft ob die angesagte Trumpffarbe richtig.

## Parameter

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

## Rückgabe

true falls die Farbe in Ordnung ist, false wenn nicht

## 5.57 Ruleset.Colour Enum-Referenz

## Öffentliche Attribute

- NONE
- HEART
- CLUB
- SPADE
- DIAMOND
- BLUE
- RED
- YELLOW

## 5.57.1 Ausführliche Beschreibung

Repräsentiert die Farbe einer Karte

## 5.58 Ruleset.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*

#### 5.58.1 Ausführliche Beschreibung

Das [GameClientUpdate](#) wird vom [RuleSet](#) ueber den [GameServer](#) an den Client geschickt und enthaelt alle Aenderungen des [GameState](#), die für den Client relevant sind. Das waeren seine Spielhand, der Ablagestapel sowie die Otherdata von allen Spielern und die Trumpfkarte.

#### 5.58.2 Beschreibung der Konstruktoren und Destruktoren

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

#### 5.58.3 Dokumentation der Elementfunktionen

5.58.3.1 [List](#)<[Card](#)> [Ruleset.GameClientUpdate.getOwnHand](#) ( ) [protected]

Holt die Karten die der Client auf der Hand hat.

#### Rückgabe

*ownHand* Die Hand des Clients

**5.58.3.2 Map<String, Card> Ruleset.GameClientUpdate.getPlayedCards ( ) [protected]**

Holt die gespielten Karten auf dem Ablagestapel.

**Rückgabe**

discardPile Die gespielten Karten

Benutzt Ruleset.GameClientUpdate.discardPile.

**5.58.3.3 OtherData Ruleset.GameClientUpdate.getOwnData ( ) [protected]**

Holt die Otherdata des Client als String als Stringrepräsentation.

**Rückgabe**

ownData Die Otherdata des Clients

**5.58.3.4 OtherData Ruleset.GameClientUpdate.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

**5.58.3.5 PlayerState Ruleset.GameClientUpdate.getCurrentPlayer ( ) [protected]**

Gibt den Spieler der momentan am Zug ist zurück.

**Rückgabe**

Der momentane Spieler

Benutzt Ruleset.GameClientUpdate.currentPlayer.

**5.58.3.6 Card Ruleset.GameClientUpdate.getTrumpCard ( ) [protected]**

Holt die aufgedeckte Trumpfkarte.

**Rückgabe**

trumpCard Die Trumpfkarte

Benutzt Ruleset.GameClientUpdate.trumpCard.

**5.59 Ruleset.GamePhase Enum-Referenz****Öffentliche Attribute**

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

### 5.59.1 Ausführliche Beschreibung

Die [GamePhase](#) modelliert die verschiedenen Zustände des Spiels im [GameState](#)

## 5.60 Ruleset.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](#) ([PlayerState](#) player)
- void [shuffleDeck](#) ()
- boolean [dealCards](#) (int number)
- void [nextPlayer](#) ()
- boolean [giveACard](#) ([PlayerState](#) player, [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](#)

### 5.60.1 Ausführliche Beschreibung

Das [GameState](#) modelliert einen aktuellen Spielzustand, es wird vom [GameServer](#) instanziiert und vom [RuleSet](#) bearbeitet. Es enthält die einzelnen [PlayerStates](#), sowie Informationen zum Ablage-, Aufnahmestapel, Rundenanzahl, den momentan aktiven Spieler sowie [GamePhase](#).

### 5.60.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.60.2.1 Ruleset.GameState.GameState ( RulesetType ruleset, List< Card > deck ) [protected]

Erstellt eine [GameState](#)klasse und [trumpCard](#) wird als [WizardCard.Empty](#) instanziiert.



## Parameter

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

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

## 5.60.3 Dokumentation der Elementfunktionen

5.60.3.1 boolean Ruleset.GameState.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 Ruleset.GameState.players und Ruleset.GameState.ruleset.

5.60.3.2 void Ruleset.GameState.setFirstPlayer ( PlayerState *player* ) [protected]

Setzt einen neuen Spieler als firstPlayer.

## Parameter

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

Benutzt Ruleset.GameState.firstPlayer.

## 5.60.3.3 PlayerState Ruleset.GameState.getFirstPlayer ( ) [protected]

Holt den Spieler der als erster am Zug war.

## Rückgabe

firstPlayer Der Spielzustand des Spielers der als erster am Zug war

Benutzt Ruleset.GameState.firstPlayer.

5.60.3.4 boolean Ruleset.GameState.setCurrentPlayer ( PlayerState *player* ) [protected]

Setzt einen neuen Spieler als currentPlayer.

## Parameter

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

Benutzt Ruleset.GameState.currentPlayer.

## 5.60.3.5 PlayerState Ruleset.GameState.getCurrentPlayer ( ) [protected]

Holt den Spieler der momentan am Zug ist.

## Rückgabe

currentPlayer Der Spielzustand des Spielers der grad am Zug ist

Benutzt Ruleset.GameState.currentPlayer.

#### 5.60.3.6 List<Card> Ruleset.GameState.getCardsLeftInDeck ( ) [protected]

Holt die Karten die noch im Aufnahmestapel sind.

##### Rückgabe

deck Holt die Karten die noch im Aufnahmestapel sind

#### 5.60.3.7 Map<String,Card> Ruleset.GameState.getPlayedCards ( ) [protected]

Holt die gespielten Karten im Ablagestapel.

##### Rückgabe

discardPile Die gespielten Karten

Benutzt Ruleset.GameState.discardPile.

#### 5.60.3.8 PlayerState Ruleset.GameState.getPlayerState ( String name ) [protected]

Holt einen bestimmten Spieler.

##### Parameter

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

##### Rückgabe

player Der Spielzustand des Spielers

Benutzt Ruleset.GameState.players.

#### 5.60.3.9 void Ruleset.GameState.setTrumpCard ( Card trumpCard ) [protected]

Setzt die Trumpfkarte.

##### Parameter

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

Benutzt Ruleset.GameState.trumpCard.

#### 5.60.3.10 Card Ruleset.GameState.getTrumpCard ( ) [protected]

Holt die momentane Trumpfkarte im Spiel.

##### Rückgabe

trumpCard Die momentane Trumpfkarte

Benutzt Ruleset.GameState.trumpCard.

#### 5.60.3.11 int Ruleset.GameState.getRoundNumber ( ) [protected]

Holt die Anzahl an Runden.

##### Rückgabe

Die Anzahl der Runden

Benutzt Ruleset.GameState.roundNumber.

**5.60.3.12** `int Ruleset.GameState.getNumberOfPlayedCards ( )` [protected]

Holt die Anzahl der gespielten Karten.

**Rückgabe**

Die Anzahl der gespielten Karten

**5.60.3.13** `List<Card> Ruleset.GameState.getPlayerCards ( PlayerState player )` [protected]

Holt die Karten eines Spielers.

**Parameter**

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

**Rückgabe**

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

Benutzt Ruleset.GameState.players.

**5.60.3.14** `boolean Ruleset.GameState.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 Ruleset.GameState.deck und Ruleset.GameState.players.

**5.60.3.15** `boolean Ruleset.GameState.giveACard ( PlayerState player, 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

**5.60.3.16** `boolean Ruleset.GameState.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

## 5.61 Ruleset.HeartsCard Enum-Referenz

Abgeleitet von [Ruleset.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`

### 5.61.1 Ausführliche Beschreibung

Modelliert eine Heartskarte.

### 5.61.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.61.2.1 Ruleset.HeartsCard.HeartsCard ( int value, Colour colour ) [private]

Erzeugt eine Heartskarte mit einem Wert und einer Farbe.

#### Parameter

<i>value</i>	Der Wert der Karte
<i>colour</i>	Die Farbe der Karte

### 5.61.3 Dokumentation der Elementfunktionen

#### 5.61.3.1 int Ruleset.HeartsCard.getValue ( )

Gibt den Wert der Karte zurück.

#### Rückgabe

Der Wert der Karte

Implementiert [Ruleset.Card](#).

### 5.61.3.2 Colour Ruleset.HeartsCard.getColour ( )

Gibt die Farbe der Karte zurück.

Rückgabe

Die Farbe der Karte

Implementiert [Ruleset.Card](#).

## 5.62 Ruleset.HeartsData Klassenreferenz

Abgeleitet von [Ruleset.OtherData](#).

Öffentliche Methoden

- String [toString](#) ( )

Geschützte Methoden

- [HeartsData](#) ( )

### 5.62.1 Ausführliche Beschreibung

Die Otherdata eines Spielers zum Spiel Hearts

## 5.63 Ruleset.OtherData Klassenreferenz

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

### 5.63.1 Ausführliche Beschreibung

[OtherData](#) speichert alle Spielinformationen eines Spielers, außer seiner eigenen Spielhand.

### 5.63.2 Dokumentation der Elementfunktionen

#### 5.63.2.1 void Ruleset.OtherData.madeTrick ( Set< Card > tricks ) [protected]

Gibt dem Spieler die Stichkarten die er gemacht.

Parameter

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

#### 5.63.2.2 List<Card> Ruleset.OtherData.removeTricks ( ) [protected]

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

Rückgabe

Die Kartenstiche

Benutzt Ruleset.OtherData.madeTricks.

#### 5.63.2.3 int Ruleset.OtherData.getNumberOfTricks ( ) [protected]

Gibt die Anzahl der gemachten Stiche des Spielers zurueck.

Rückgabe

Die Anzahl der gemachten Stiche

#### 5.63.2.4 void Ruleset.OtherData.setPoints ( int points ) [protected]

Setzt den Punktestand eines Spielers.

Parameter

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

Benutzt Ruleset.OtherData.points.

#### 5.63.2.5 int Ruleset.OtherData.getPoints ( ) [protected]

Gibt den Punktestand eines Spielers zurück.

Rückgabe

Der Punktestand

Benutzt Ruleset.OtherData.points.

## 5.64 Ruleset.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](#)

## 5.64.1 Ausführliche Beschreibung

Repraesentiert den Spielzustand eines Spielers, und wird unter anderem im [GameState](#) gespeichert. Sie enthält den Namen des Spielers, seine Handkarten und [OtherData](#).

## 5.64.2 Beschreibung der Konstruktoren und Destruktoren

5.64.2.1 Ruleset.PlayerState.PlayerState ( String *name*, RulesetType *ruleset* )

Erstellt einen [PlayerState](#).

## Parameter

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

Benutzt Ruleset.RulesetType.Hearts, Ruleset.PlayerState.name und Ruleset.PlayerState.otherData.

## 5.64.3 Dokumentation der Elementfunktionen

## 5.64.3.1 String Ruleset.PlayerState.getName ( ) [protected]

Holt den Namen eines Spielers.

## Rückgabe

*name* Der Name des Spielers

## 5.64.3.2 List&lt;Card&gt; Ruleset.PlayerState.getHand ( ) [protected]

Holt die Kartenhand des Spielers.

## Rückgabe

*ownHand* Die Kartenhand des Spielers

## 5.64.3.3 OtherData Ruleset.PlayerState.getOtherData ( ) [protected]

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

## Rückgabe

*otherData* Die [OtherData](#) eines Spielers

Benutzt Ruleset.PlayerState.otherData.

Wird benutzt von Ruleset.ServerRuleset.getPoints().

5.64.3.4 void Ruleset.PlayerState.addCard ( Card *card* ) [protected]

Gibt dem Spieler eine Karte.

## Parameter

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

5.64.3.5 `boolean Ruleset.PlayerState.removeCard ( Card card )` [protected]

Entfernt eine Karte aus der Hand des Spielers.

## Parameter

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

## Rückgabe

Gibt true zurueck wenn die Karte in der Hand ist und false sonst

5.65 `Ruleset.RulesetType` Enum-Referenz

## Öffentliche Attribute

- [Wizard](#)
- [Hearts](#)

## 5.65.1 Ausführliche Beschreibung

Den Regelwerk Typ den ein Spiel hat.

5.66 `Ruleset.ServerHearts` Klassenreferenz

Abgeleitet von [Ruleset.ServerRuleset](#).

## Öffentliche Methoden

- [ServerHearts](#) ([GameServer](#) s)
- void [resolveMessage](#) ([MsgMultiCards](#) msgMultiCard, String name)

## Geschützte Methoden

- boolean [isValidMove](#) ([Card](#) card)
- int [getEndingPoints](#) ()
- void [calculateRoundOutcome](#) ()
- void [calculateTricks](#) ()
- String [getWinner](#) ()
- [GameClientUpdate](#) [generateGameClientUpdate](#) (String player)

## Private Methoden

- boolean [areValidChosenCards](#) (Set< [Card](#) > cards, String name)

## Statische, private Attribute

- static final int [MIN\\_PLAYERS](#) = 4
- static final int [MAX\\_PLAYERS](#) = 4
- static final [RulesetType](#) [RULESET](#) = [RulesetType.Hearts](#)
- static final int [ENDING\\_POINTS](#) = 100



## 5.66.1 Ausführliche Beschreibung

Diese Klasse erstellt das Regelwerk zum Spiel Hearts. Sie enthaelt zudem weitere Methoden, welche für das Spiel Hearts spezifisch benoetigt werden, wie die Regelung zum Tausch von Karten und die Berechnung der Stichpunkten.

## 5.66.2 Dokumentation der Elementfunktionen

5.66.2.1 `boolean Ruleset.ServerHearts.isValidMove ( Card card ) [protected],[virtual]`

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

## Parameter

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

## Rückgabe

true falls Zug gueltig und false wenn nicht

Implementiert [Ruleset.ServerRuleset](#).

5.66.2.2 `int Ruleset.ServerHearts.getEndingPoints ( ) [protected]`

Holt die Anzahl der Punkte die ein Spieler haben kann ab der, das Spiel vorbei ist.

## Rückgabe

Anzahl der maximalen Punkte

Benutzt `Ruleset.ServerHearts.ENDING_POINTS`.

5.66.2.3 `void Ruleset.ServerHearts.resolveMessage ( MsgMultiCards msgMultiCard, String name )`

Verarbeitet die `RulesetMessage` dass mehrerer Karten vom Spieler uebergeben werden.

Die wird dann in `areValidChosenCards` ueberprueft, bei falsche Eingabe wird eine `MsgMultiCardsRequest` an den selben Spieler gesendet. Bei richtiger Eingabe geht das Spiel weiter.

## Parameter

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

5.66.2.4 `boolean Ruleset.ServerHearts.areValidChosenCards ( Set< Card > cards, String name ) [private]`

Ueberprueft ob eine uebergebenes Kartenset von einem Spieler gültig ist.

## Parameter

<i>cards</i>	Ein Kartenset
--------------	---------------

## Rückgabe

true falls das Kartenset gueltig ist, false wenn nicht

5.66.2.5 `GameClientUpdate Ruleset.ServerHearts.generateGameClientUpdate ( String player ) [protected],[virtual]`

Erzeugt ein [GameClientUpdate](#) welches individuell für jeden Benutzer ist.

## Parameter

<i>player</i>	Dem Spieler
---------------	-------------

Implementiert [Ruleset.ServerRuleset](#).

## 5.67 Ruleset.ServerRuleset Klassenreferenz

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

### Öffentliche Methoden

- [ServerRuleset](#) ([RulesetType](#) ruleset, int min, int max, [GameServer](#) server)
- [RulesetType](#) getRulesetType ()
- int getMinPlayers ()
- int getMaxPlayers ()
- [GamePhase](#) getGamePhase ()
- void runGame ()
- void resolveMessage ([RulesetMessage](#) message, String name)
- void resolveMessage ([MsgCard](#) msgCard, String name)

### Geschützte Methoden

- int getRoundNumber ()
- void setGamePhase ([GamePhase](#) phase)
- 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 ([PlayerState](#) player)
- void send ([RulesetMessage](#) message, String name)
- void broadcast ([RulesetMessage](#) message)
- void setPoints ([PlayerState](#) player, int i)
- int getPoints ([PlayerState](#) player)
- boolean dealCards (int number)
- boolean giveACard ([PlayerState](#) player, [Card](#) card)
- boolean playCard ([Card](#) card)
- void setTrumpCard ([Card](#) card)
- abstract boolean isValidMove ([Card](#) card)
- abstract void calculateTricks ()
- abstract void calculateRoundOutcome ()
- abstract String getWinner ()
- abstract [GameClientUpdate](#) generateGameClientUpdate (String player)

### Private Methoden

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

## Private Attribute

- [GameServer](#) server
- [GameState](#) gameState
- [GamePhase](#) gamePhase
- final [RulesetType](#) RULESET
- final int [MIN\\_PLAYERS](#)
- final int [MAX\\_PLAYERS](#)

## 5.67.1 Ausführliche Beschreibung

Das [ServerRuleset](#) ist eine abstrakte Klasse und fuer den Ablauf und die Einhaltung der Regeln eines Spiels zustaeendig. Das [ServerRuleset](#) wird im [GameServer](#) instanziiert und verwaltet die Zustaeende des [GameStates](#) im Server. Mit der Methode [isValidMove\(\)](#) wird eine Eingabe eines Clients auf Regelkonformität ueberprueft und dann das [GameState](#) veraendert. Ueber [resolveMessage\(\)](#) kann eine [GameServer](#)instanz eine [RulesetMessage](#) vom Player an das Ruleset weiterleiten.

## 5.67.2 Beschreibung der Konstruktoren und Destruktoren

## 5.67.2.1 Ruleset.ServerRuleset.ServerRuleset ( RulesetType ruleset, int min, int max, GameServer server )

Erstellt ein [ServerRuleset](#) und erzeugt ein [GameState](#).

## 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 [Ruleset.ServerRuleset.createDeck\(\)](#), [Ruleset.ServerRuleset.gamePhase](#), [Ruleset.ServerRuleset.gameState](#), [Ruleset.ServerRuleset.MAX\\_PLAYERS](#), [Ruleset.ServerRuleset.MIN\\_PLAYERS](#), [Ruleset.ServerRuleset.RULESET](#), [Ruleset.ServerRuleset.server](#) und [Ruleset.GamePhase.Start](#).

## 5.67.3 Dokumentation der Elementfunktionen

## 5.67.3.1 int Ruleset.ServerRuleset.getRoundNumber ( ) [protected]

Holt die aktuelle Rundenanzahl zurueck.

## Rückgabe

Die aktuelle Rundenanzahl

## 5.67.3.2 RulesetType Ruleset.ServerRuleset.getRulesetType ( )

Gibt den Typ des Regelwerks zurueck.

## Rückgabe

Der Typ vom Regelwerk

Benutzt [Ruleset.ServerRuleset.RULESET](#).

### 5.67.3.3 `int Ruleset.ServerRuleset.getMinPlayers ( )`

Gibt die Mindestanzahl an Spielern zurueck für dieses Spiel.

#### Rückgabe

Die Mindestanzahl an Spielern

Benutzt Ruleset.ServerRuleset.MIN\_PLAYERS.

### 5.67.3.4 `int Ruleset.ServerRuleset.getMaxPlayers ( )`

Gibt die Maximale Anzahl an Spielern zurueck.

#### Rückgabe

Die maximale Anzahl an Spielern

Benutzt Ruleset.ServerRuleset.MAX\_PLAYERS.

### 5.67.3.5 `void Ruleset.ServerRuleset.setGamePhase ( GamePhase phase )` [protected]

Aendert den momentanen Spielphase.

#### Parameter

<i>phase</i>	Die neue Spielphase
--------------	---------------------

### 5.67.3.6 `GamePhase Ruleset.ServerRuleset.getGamePhase ( )`

Gibt den momentanen Spielzustand zurueck.

#### Rückgabe

Gibt die momentan Spielphase zurueck

Benutzt Ruleset.ServerRuleset.gamePhase.

### 5.67.3.7 `List<Card> Ruleset.ServerRuleset.createDeck ( )` [private]

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

#### Rückgabe

Gibt ein Kartendeck zurueck

Benutzt Ruleset.ServerRuleset.RULESET.

Wird benutzt von Ruleset.ServerRuleset.ServerRuleset().

### 5.67.3.8 `void Ruleset.ServerRuleset.setFirstPlayer ( PlayerState player )` [protected]

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

#### Parameter

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

### 5.67.3.9 `PlayerState Ruleset.ServerRuleset.getFirstPlayer ( )` [protected]

Holt den Spieler der als erster am Zug war.

**Rückgabe**

firstPlayer Der Spielzustand des Spielers der als erster am Zug war

**5.67.3.10 boolean Ruleset.ServerRuleset.nextPlayer ( ) [protected]**

Setzt den naechsten Spieler in der List als currentPlayer.

**Rückgabe**

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

**5.67.3.11 boolean Ruleset.ServerRuleset.setCurrentPlayer ( PlayerState player ) [protected]**

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

**Parameter**

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

**Rückgabe**

false wenn der selbe Spieler nochmal als currentPlayer gesetzt wird

**5.67.3.12 OtherData Ruleset.ServerRuleset.getOtherData ( PlayerState player ) [protected]**

Die [OtherData](#) eines Spielers.

**Parameter**

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

**Rückgabe**

Gibt [OtherData](#) zurück

**5.67.3.13 PlayerState Ruleset.ServerRuleset.getCurrentPlayer ( ) [protected]**

Holt den Spieler der gerade am Zug ist.

**Rückgabe**

currentPlayer Der Spielzustand des Spielers der grad am Zug ist

**5.67.3.14 void Ruleset.ServerRuleset.addPlayerToGame ( String name ) [protected]**

Fuegt einen Spieler ins Spiel ein.

**Parameter**

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

**5.67.3.15 PlayerState Ruleset.ServerRuleset.getPlayerState ( String name ) [protected]**

Holt den Spielerzustand eines Spielers.

## Parameter

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

## Rückgabe

playerState Spielzustand eines Spielers

#### 5.67.3.16 List<Card> Ruleset.ServerRuleset.getPlayerCards ( PlayerState *player* ) [protected]

Holt die Spielkarten eines Spielers.

## Parameter

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

## Rückgabe

Die Spielkarten des Spielers

#### 5.67.3.17 void Ruleset.ServerRuleset.send ( RulesetMessage *message*, String *name* ) [protected]

Schickt eine Nachricht an einen Spieler, über den Gameserver.

## Parameter

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

#### 5.67.3.18 void Ruleset.ServerRuleset.broadcast ( RulesetMessage *message* ) [protected]

Schickt eine Nachricht an alle Spieler.

## Parameter

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

#### 5.67.3.19 void Ruleset.ServerRuleset.resolveMessage ( RulesetMessage *message*, String *name* )

Verarbeitet eine allgemeine RulesetMessage.

## Parameter

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

#### 5.67.3.20 void Ruleset.ServerRuleset.resolveMessage ( MsgCard *msgCard*, String *name* )

Verarbeitet die RulesetMessage dass eine Karte vom Spieler gespielt.

Die wird dann in isValidMove überprüft, bei falsche Eingabe wird eine MsgCardRequest an den selben Spieler geschickt. Bei richtiger Eingabe geht das Spiel weiter.

## Parameter

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

#### 5.67.3.21 void Ruleset.ServerRuleset.setPoints ( PlayerState *player*, int *i* ) [protected]

Setzt den Punktestand eines Spielers.

## Parameter

<i>player</i>	Der Spieler
<i>i</i>	Punktestand

5.67.3.22 int Ruleset.ServerRuleset.getPoints ( PlayerState *player* ) [protected]

Holt den Punktestand eines Spielers.

## Parameter

<i>player</i>	Der Spieler
---------------	-------------

## Rückgabe

Die Punkte des Spielers

Benutzt Ruleset.PlayerState.getOtherData().

5.67.3.23 boolean Ruleset.ServerRuleset.dealCards ( int *number* ) [protected]

Verteilt eine bestimmte Anzahl an Karten an die Spieler.

## Parameter

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

## Rückgabe

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

5.67.3.24 boolean Ruleset.ServerRuleset.giveACard ( PlayerState *player*, Card *card* ) [protected]

Gibt einem Spieler eine bestimmte Karte.

## Parameter

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

## Rückgabe

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

5.67.3.25 boolean Ruleset.ServerRuleset.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, false wenn nicht

5.67.3.26 void Ruleset.ServerRuleset.setTrumpCard ( Card *card* ) [protected]

Setzt eine Karte als Trumpf.

## Parameter

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

5.67.3.27 **abstract boolean** Ruleset.ServerRuleset.isValidMove ( **Card** *card* ) [protected],[pure virtual]

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

## Parameter

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

## Rückgabe

true falls Zug gueltig und false wenn nicht

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

5.67.3.28 **abstract GameClientUpdate** Ruleset.ServerRuleset.generateGameClientUpdate ( **String** *player* ) [protected],[pure virtual]

Erzeugt ein [GameClientUpdate](#) welches individuell für jeden Benutzer ist.

## Parameter

<i>player</i>	Dem Spieler
---------------	-------------

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

## 5.68 Ruleset.ServerWizard Klassenreferenz

Abgeleitet von [Ruleset.ServerRuleset](#).

## Öffentliche Methoden

- [ServerWizard](#) ([GameServer](#) s)
- void [resolveMessage](#) ([MsgNumber](#) msgNumber, **String** name)
- void [resolveMessage](#) ([MsgSelection](#) msgSelection, **String** name)

## Geschützte Methoden

- boolean [isValidMove](#) ([Card](#) card)
- void [calculateRoundOutcome](#) ()
- int [getplayingRounds](#) ()
- void [calculateTricks](#) ()
- **String** [getWinner](#) ()
- [GameClientUpdate](#) [generateGameClientUpdate](#) (**String** player)

## Private Methoden

- void [setPlayingRounds](#) (int rounds)
- boolean [isValidNumber](#) (int number, **String** name)
- boolean [isValidColour](#) ([Colour](#) colour, **String** name)



## Private Attribute

- int [playingRounds](#)

## Statische, private Attribute

- static final int [MIN\\_PLAYERS](#) = 3
- static final int [MAX\\_PLAYERS](#) = 6
- static final [RulesetType RULESET](#) = [RulesetType.Wizard](#)

## 5.68.1 Ausführliche Beschreibung

Diese Klasse erstellt das Regelwerk zum Spiel Wizard. Sie enthaelt zudem weitere Methoden, welche für das Spiel Wizard spezifisch benoetigt werden, wie das Ansage von Stichen, der Bestimmung von Trumpffarben und die Bestimmung der Rundenanzahl.

## 5.68.2 Dokumentation der Elementfunktionen

## 5.68.2.1 boolean Ruleset.ServerWizard.isValidMove ( Card card ) [protected], [virtual]

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

## Parameter

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

## Rückgabe

true falls Zug gueltig und false wenn nicht

Implementiert [Ruleset.ServerRuleset](#).

## 5.68.2.2 void Ruleset.ServerWizard.setPlayingRounds ( int rounds ) [private]

Setzt die Anzahl an Runden die es in diesem Spiel gibt.

## Parameter

<i>rounds</i>	Die Anzahl an Runden
---------------	----------------------

Benutzt Ruleset.ServerWizard.playingRounds.

## 5.68.2.3 int Ruleset.ServerWizard.getplayingRounds ( ) [protected]

Holt die Anzahl der Runden die gespielt werden.

## Rückgabe

playingRounds Die Anzahl an Runden

Benutzt Ruleset.ServerWizard.playingRounds.

## 5.68.2.4 void Ruleset.ServerWizard.resolveMessage ( MsgNumber msgNumber, String name )

Verarbeitet die RulesetMessage dass der Spieler eine Stichansage macht.

Die wird dann in isValidNumber überprüft, bei falsche Eingabe wird eine MsgCardRequest an den selben Spieler geschickt. Bei richtiger Eingabe geht das Spiel weiter.

## Parameter

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

5.68.2.5 `boolean Ruleset.ServerWizard.isValidNumber ( int number, String name )` [private]

Ueberprueft ob eine eingegebene Stichangabe eines Spielers gueltig ist.

## Parameter

<i>number</i>	Die Stichangabe
<i>name</i>	Der Name vom Spieler

## Rückgabe

true falls die Stichangabe gültig ist, false wenn nicht

5.68.2.6 `void Ruleset.ServerWizard.resolveMessage ( MsgSelection msgSelection, String name )`

Verarbeitet die RulesetMessage dass mehrerer Karten vom Spieler uebergeben werden.

Die wird dann in isValidColour überprüft, bei falsche Eingabe wird MsgMultiCardRequest an den selben Spieler geschickt. Bei richtiger Eingabe geht das Spiel weiter.

## Parameter

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

5.68.2.7 `boolean Ruleset.ServerWizard.isValidColour ( Colour colour, String name )` [private]

Ueberprueft ob eine eingebene Trumpffarbe eines Spielers gueltig ist.

## Parameter

<i>colour</i>	Die Trumpffarbe
<i>name</i>	Der Name des Spielers

## Rückgabe

true falls die Farbe gueltig ist, false wenn nicht

5.68.2.8 `GameClientUpdate Ruleset.ServerWizard.generateGameClientUpdate ( String player )` [protected], [virtual]

Erzeugt ein [GameClientUpdate](#) welches individuell für jeden Benutzer ist.

## Parameter

<i>player</i>	Dem Spieler
---------------	-------------

Implementiert [Ruleset.ServerRuleset](#).

## 5.68.3 Dokumentation der Datenelemente

5.68.3.1 `int Ruleset.ServerWizard.playingRounds` [private]

Die Anzahl an Runden die gespielt wird.

Ist abhaengig von der Spieleranzahl.

Wird benutzt von Ruleset.ServerWizard.getplayingRounds() und Ruleset.ServerWizard.setPlayingRounds().

## 5.69 Ruleset.WizardCard Enum-Referenz

Abgeleitet von [Ruleset.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`

### 5.69.1 Ausführliche Beschreibung

Modelliert eine Wizardkarte

### 5.69.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.69.2.1 Ruleset.WizardCard.WizardCard ( int value, Colour colour ) [private]

Erzeugt eine Wizardkarte.

#### Parameter

<i>value</i>	Der Wert der Karte
<i>colour</i>	Die Farbe der Karte

### 5.69.3 Dokumentation der Elementfunktionen

#### 5.69.3.1 int Ruleset.WizardCard.getValue ( )

Gibt den Wert der Karte zurück.

#### Rückgabe

Der Wert der Karte

Implementiert [Ruleset.Card](#).

### 5.69.3.2 Colour Ruleset.WizardCard.getColour ( )

Gibt die Farbe der Karte zurück.

#### Rückgabe

Die Farbe der Karte

Implementiert [Ruleset.Card](#).

## 5.70 Ruleset.WizData Klassenreferenz

Abgeleitet von [Ruleset.OtherData](#).

#### Öffentliche Methoden

- String [toString](#) ( )

#### Geschützte Methoden

- [WizData](#) ( )
- int [getAnnouncedTricks](#) ( )
- void [setAnnouncedTricks](#) (int *annouceTricks*)

#### Private Attribute

- int [announcedTricks](#)

### 5.70.1 Ausführliche Beschreibung

Die Otherdata eines Spielers zum Spiel Wizard

### 5.70.2 Dokumentation der Elementfunktionen

#### 5.70.2.1 int Ruleset.WizData.getAnnouncedTricks ( ) [protected]

Holt die angesagten Stiche des Spielers.

#### Rückgabe

*announcedTricks* Die angesagten Stiche

#### 5.70.2.2 void Ruleset.WizData.setAnnouncedTricks ( int *annouceTricks* ) [protected]

Beim Spielstart werden die vorausgesagten Stiche des Spieler gespeichert.

#### Parameter

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

## 5.71 Server.GameServer Klassenreferenz

Abgeleitet von [Server.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 [sendRulesetMessage](#) (String player, RulesetMessage message)
- void [broadcastRulesetMessage](#) (RulesetMessage message)
- 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

#### 5.71.1 Ausführliche Beschreibung

Diese Klasse ist fuer die Verwaltung eines Spieles zustaendig. Sie verwaltet die Kommunikation zwischen den Clients waehrend 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.

#### 5.71.2 Beschreibung der Konstruktoren und Destruktoren

##### 5.71.2.1 [Server.GameServer.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 uebergebenen Werte. Setzt den gameMasterName auf den Namen des gameMaster und fuegt 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

### 5.71.3 Dokumentation der Elementfunktionen

#### 5.71.3.1 **GameServerRepresentation** `Server.GameServer.getRepresentation ( )`

Erstellt eine neue [GameServerRepresentation](#) und gibt sie zurueck.

##### Rückgabe

Gibt die neue [GameServerRepresentation](#) zurueck

Benutzt `Server.GameServer.currentPlayers`, `Server.GameServer.gameMasterName`, `Server.GameServer.hasPassword`, `Server.GameServer.maxPlayers`, `Server.GameServer.name` und `Server.GameServer.rulesetType`.

#### 5.71.3.2 `synchronized void Server.GameServer.addPlayer ( Player player )`

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

Es wird vorausgesetzt, dass der [Player](#) gueltig und noch nicht im Set vorhanden ist. Zusaetzlich wird die Zahl der `currentPlayers` um eins Erhoeht.

##### Parameter

<i>player</i>	ist der <a href="#">Player</a> , der hinzugefuegt wird
---------------	--

Benutzt `Server.GameServer.currentPlayers`.

#### 5.71.3.3 `synchronized void Server.GameServer.removePlayer ( Player player )`

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

Es wird vorausgesetzt, dass der [Player](#) gueltig und im Set vorhanden ist. Zusaetzlich wird die Zahl der `currentPlayers` um eins Verringert.

##### Parameter

<i>player</i>	ist der <a href="#">Player</a> , der entfernt wird
---------------	--

Benutzt `Server.GameServer.currentPlayers`.

#### 5.71.3.4 `void Server.GameServer.sendRulesetMessage ( String player, RulesetMessage message )`

Diese Methode verpackt eine `RulesetMessage` in ein `ComObject` und verschickt es mit `sendToPlayer()` an einen bestimmten Spieler.

##### Parameter

<i>player</i>	ist der Name des Spielers an den die Nachricht verschickt wird
<i>message</i>	ist die Ruleset Nachricht, die in ein <code>ComObject</code> verpackt wird

Benutzt `Server.Server.sendToPlayer()`.

#### 5.71.3.5 `void Server.GameServer.broadcastRulesetMessage ( RulesetMessage message )`

Diese Methode verpackt eine `RulesetMessage` in ein `ComObject` und verschickt es mit `broadcast()` an alle Spieler.

## Parameter

<i>message</i>	ist die Ruleset Nachricht, die in ein ComObject verpackt wird
----------------	---

Benutzt Server.Server.broadcast().

5.71.3.6 void Server.GameServer.receiveMessage ( Player *player*, ComKickPlayerRequest *kickPlayer* )

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

Der [Player](#) wird durch Aufruf von changeServer an die Lobby zurueckgegeben. An diesen Spieler wird ein ComInitLobby und ein ComWarning geschickt. Danach wird ein ComUpdatePlayerlist Objekt mit broadcast an alle Clients im Spiel verschickt.

## Parameter

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

5.71.3.7 void Server.GameServer.receiveMessage ( Player *player*, ComChatMessage *chat* )

Diese Methode ist dafur zustandig eine Chatnachricht an alle Clients im Spiel zu verschicken.

Dafuer 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 enthaelt

Benutzt Server.Server.broadcast().

5.71.3.8 void Server.GameServer.receiveMessage ( Player *player*, ComClientLeave *leave* )

Diese Methode gibt einen [Player](#), der die GameLobby verlassen will, durch Aufruf von changeServer an die ServerLobby zurueck 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 zurueckkehrt

5.71.3.9 void Server.GameServer.receiveMessage ( Player *player*, ComClientQuit *quit* )

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

Alle Spieler, die sich im Spiel befinden werden durch Aufruf von changeServer an die Lobby zurueckgegeben 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 verlaesst

5.71.3.10 void Server.GameServer.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

**5.71.3.11 void Server.GameServer.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

**5.71.3.12 ComInitGameLobby Server.GameServer.initLobby ( )**

Baut ein neues ComInitGameLobby Objekt und gibt es zurueck.

**Rückgabe**

Gibt das ComInitGameLobby Objekt zurueck

**5.71.3.13 void Server.GameServer.handleIOException ( Player *player* )**

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

Der uebergebene [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 zurueckgegeben 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
---------------	---

**5.71.4 Dokumentation der Datenelemente****5.71.4.1 String Server.GameServer.password [private]**

Das Passwort, das der Spielleiter beim erstellen gesetzt hat.

Ist NULL, falls es kein Passwort gibt.

**5.72 Server.GameServerRepresentation Klassenreferenz****Öffentliche Methoden**

- [GameServerRepresentation](#) (String gameMaster, String gameName, int max, int current, [RulesetType](#) type, boolean password)
- String [getGameMasterName](#) ()
- String [getName](#) ()
- int [getMaxPlayers](#) ()
- int [getCurrentPlayers](#) ()
- [RulesetType](#) [getRuleset](#) ()
- boolean [isHasPassword](#) ()



## Private Attribute

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

## 5.72.1 Ausführliche Beschreibung

Dies ist eine Klasse, die Informationen ueber den Zustand eines Spielservers bereithaelt. Sie wird dem ComObjekt [ComLobbyUpdateGameList](#) angehaengt, um die Spielliste in der [GameLobby](#) aktualisieren zu koennen.

## 5.72.2 Beschreibung der Konstruktoren und Destruktoren

5.72.2.1 [Server.GameServerRepresentation.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 moegliche 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

## 5.73 Server.LobbyServer Klassenreferenz

Abgeleitet von [Server.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

### 5.73.1 Ausführliche Beschreibung

Diese Klasse ist fuer die Verwaltung der Spiellobby auf dem [Server](#) verantwortlich. Sie erstellt neue Spiele und verwaltet laufende Spiele. Auch wird der Chatverkehr ueber sie an die verbundenen Spieler weitergeleitet. Die LobbyServer-Klasse erbt Methoden zur Kommunikation vom [Server](#).

### 5.73.2 Dokumentation der Elementfunktionen

#### 5.73.2.1 void Server.LobbyServer.addName ( String *name* )

Fuegt 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 eingefuegt wird
-------------	----------------------------------

#### 5.73.2.2 void Server.LobbyServer.removeName ( String *name* )

Loescht einen Benutzennamen aus dem Namensset.

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

#### Parameter

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

#### 5.73.2.3 void Server.LobbyServer.addGameServer ( [GameServer](#) *game* )

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

#### Parameter

<i>game</i>	ist der <a href="#">GameServer</a> der eingefuegt wird
-------------	--

#### 5.73.2.4 void Server.LobbyServer.removeGameServer ( [GameServer](#) *game* )

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

#### Parameter

<i>game</i>	ist der <a href="#">GameServer</a> der geloescht wird
-------------	---

#### 5.73.2.5 void Server.LobbyServer.receiveMessage ( [Player](#) *player*, [ComChatMessage](#) *chat* )

Diese ueberladene Methode ist dafuer zustaeendig eine Chatnachricht an alle Clients im Spiel zu verschicken.

Dafuer 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 enthaelt

Benutzt Server.Server.broadcast().

5.73.2.6 void Server.LobbyServer.receiveMessage ( Player *player*, ComClientQuit *quit* )

Diese ueberladene 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 vollstaendig verlaesst

5.73.2.7 void Server.LobbyServer.receiveMessage ( Player *player*, ComCreateGameRequest *create* )

Diese ueberladene Methode erstellt einen neuen [GameServer](#) fuegt ihm den [Player](#) durch Aufruf von dessen changeServer Methode hinzu.

Der neue [GameServer](#) wird in das gameServerSet eingefuegt. Durch broadcast wird im [LobbyServer](#) sowohl Com-UpdatePlayerlist als auch ein ComLobbyUpdateGamelist verschickt. Zusaetzlich 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

5.73.2.8 void Server.LobbyServer.receiveMessage ( Player *player*, ComJoinRequest *join* )

Diese ueberladene Methode fuegt einen [Player](#) dem entsprechenden [GameServer](#) hinzu.

Falls das Passwort nicht leer ist wird geprueft, ob es mit dem Passwort des Spieles uebereinstimmt, wenn nicht, wird ein ComWarning an den Client geschickt. Ansonsten wird und der [Player](#) dem, durch Namen des Spielleiters identifizierten, Gameserver, durch Aufruf von changeServer uebergeben. 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

5.73.2.9 void Server.LobbyServer.receiveMessage ( Player *player*, ComLoginRequest *login* )

Diese ueberladene Methode ueberprueft, 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 eingefuegt. Der Name wird in das Set names eingefuegt. 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 enthaelt

#### 5.73.2.10 ComInitLobby Server.LobbyServer.initLobby ( )

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

##### Rückgabe

Gibt das ComInitLobby Objekt zurueck

#### 5.73.2.11 void Server.LobbyServer.handleIOException ( Player player )

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

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

##### Parameter

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

### 5.74 Server.LobbyServer.ClientListenerThread Klassenreferenz

Abgeleitet von Runnable.

##### Öffentliche Methoden

- [ClientListenerThread](#) ()
- void [run](#) ()

#### 5.74.1 Ausführliche Beschreibung

Diese innere Klasse ist fuer das Zustandekommen von Clientverbindungen zustaeendig. Der Thread wartet auf eingehende Clientverbindungen, stellt diese her und instanziert fuer jede Verbindung eine Klasse [Player](#). Dieser wird dann dem [LobbyServer](#) uebergeben.

### 5.75 Server.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](#)

### 5.75.1 Ausführliche Beschreibung

Die Player-Klasse wird zum Versenden von Java Serializable Objects, sowie zum Annehmen solcher verwendet. Sie verwaltet fuer die Dauer einer Serververbindung die Verbindung zu einem Client.

### 5.75.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.75.2.1 Server.Player.Player ( Server lobbyServer, ObjectOutputStream output, ObjectInputStream input )

Konstruktor des Players, in ihm werden die Attribute server, comOut und comIn mit vom ClientListenerThread uebergebenen Werten Instanziert.

##### 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 Server.Player.comIn, Server.Player.comOut und Server.Player.server.

### 5.75.3 Dokumentation der Elementfunktionen

#### 5.75.3.1 void Server.Player.run ( )

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

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

Benutzt Server.Player.comIn und Server.Player.server.

#### 5.75.3.2 void Server.Player.send ( ComObject com )

Diese Methode schickt ein ComObjekt an den Client.

Sie faengt 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
------------	---------------------------------------

#### 5.75.3.3 void Server.Player.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 eingefuegt. 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 Server.Player.getName() und Server.Player.server.

#### 5.75.3.4 String Server.Player.getName ( )

Getter-Methode fuer den Benutzernamen.

**Rückgabe**

gibt den Benutzernamen des Spielers zurueck

Benutzt Server.Player.name.

Wird benutzt von Server.Player.changeServer().

**5.75.3.5 void Server.Player.setName ( String newName )**

Setter-Methode fuer den Benutzernamen.

**Parameter**

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

Benutzt Server.Player.name.

**5.76 Server.Server Klassenreferenz**

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

**5.76.1 Ausführliche Beschreibung**

Ist eine abstrakte Klasse, von der die Klassen [LobbyServer](#) und [GameServer](#) erben. Es stellt Methoden zur Nachrichtenversendung und -verarbeitung bereit, sowie zur Verwaltung von Playern

**5.76.2 Dokumentation der Elementfunktionen****5.76.2.1 void Server.Server.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 ComObject vom Client verschickt wurde

**5.76.2.2 synchronized void Server.Server.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 uebergebenen Benutzernamens identifiziert. Es wird vorausgesetzt, dass der Name und das ComObject gueltig 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.Server.playerSet.

Wird benutzt von Server.GameServer.sendRulesetMessage().

5.76.2.3 synchronized void Server.Server.addPlayer ( [Player](#) *player* )

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

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

## Parameter

<i>player</i>	ist der <a href="#">Player</a> , der hinzugefuegt wird
---------------	--

5.76.2.4 synchronized void Server.Server.removePlayer ( [Player](#) *player* )

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

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

## Parameter

<i>player</i>	ist der <a href="#">Player</a> , der entfernt wird
---------------	--

5.76.2.5 synchronized void Server.Server.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 gueltig ist.

## Parameter

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

Benutzt Server.Server.playerSet.

Wird benutzt von Server.GameServer.broadcastRulesetMessage(), Server.LobbyServer.receiveMessage() und Server.GameServer.receiveMessage().

5.76.2.6 void Server.Server.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
---------------	---

## 5.77 Server.ServerMain Klassenreferenz

## Öffentliche, statische Methoden

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

## Private Attribute

- [LobbyServer](#) [lobbyServer](#)

### 5.77.1 Ausführliche Beschreibung

Diese Klasse startet den [Server](#) und ist fuer die Konfiguration des Servers verantwortlich.

### 5.77.2 Dokumentation der Elementfunktionen

#### 5.77.2.1 `static void Server.ServerMain.main ( String[] args ) [static]`

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

Parameter

<code>args</code>	
-------------------	--

## 6 JUnit-Tests

JUnit-Tests werden für die folgenden Klassen geschrieben: ClientModel, LobbyServer, GameServer, ClientHearts, ClientWizard, ServerHearts und ServerWizard. Für die folgenden Fälle wurden bereits Tests implementiert.

### 6.1 isValidWizardMove

---

```
package Ruleset;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import test.TestGameServer;
import test.TestLobbyServer;
import test.TestPlayer;

import Server.GameServer;
import Server.LobbyServer;
import Server.Player;

public class TestIsValidMoveWizard {

    ServerRuleset ruleset;

    GameServer gameServer;

    LobbyServer lobbyServer;

    Player player;
    String player1;
    String player2;
    String player3;
    PlayerState playerState1;
    PlayerState playerState2;
    PlayerState playerState3;

    @Before
    public void setUp() throws Exception {
        player1 = "Tick";
```

---



```

    player2 = "Trick";
    player3 = "Track";
    lobbyServer = new TestLobbyServer();
    player = new TestPlayer(lobbyServer, null, null);
    gameServer = new TestGameServer(lobbyServer, player, "Mein
        Spiel", RulesetType.Wizard,
            "", false);
    ruleset = new ServerWizard(gameServer);

    ruleset.addPlayerToGame(player1);
    ruleset.addPlayerToGame(player2);
    ruleset.addPlayerToGame(player3);

    playerState1 = ruleset.getPlayerState(player1);
    playerState2 = ruleset.getPlayerState(player2);
    playerState3 = ruleset.getPlayerState(player3);

    ruleset.setFirstPlayer(ruleset.getPlayerState(player1));
    ruleset.setTrumpCard(WizardCard.VierRot);

    ruleset.giveACard(playerState1, WizardCard.DreiGruen);
    ruleset.giveACard(playerState1, WizardCard.ZaubererRot);
    ruleset.giveACard(playerState1, WizardCard.ZweiBlau);

    ruleset.giveACard(playerState2, WizardCard.ZweiGruen);
    ruleset.giveACard(playerState2, WizardCard.DreiRot);
    ruleset.giveACard(playerState2, WizardCard.ZweiGelb);

    ruleset.giveACard(playerState3, WizardCard.NarrBlau);
    ruleset.giveACard(playerState3, WizardCard.EinsGruen);
    ruleset.giveACard(playerState3, WizardCard.ZweiRot);
}

@Test
public void testSorcerer() {
    ruleset.playCard(WizardCard.ZaubererRot);
    ruleset.setCurrentPlayer(playerState2);

    boolean isValidMove = ruleset.isValidMove(WizardCard.DreiRot);

    assertTrue(isValidMove);
}

@Test
public void testRed3OnGreen3() {
    ruleset.playCard(WizardCard.DreiGruen);
    ruleset.setCurrentPlayer(playerState2);
    boolean isValidMove = ruleset.isValidMove(WizardCard.DreiRot);

    assertFalse(isValidMove);
}

@Test
public void testGreen2OnGreen3() {
    ruleset.playCard(WizardCard.DreiGruen);
    ruleset.setCurrentPlayer(playerState2);

    boolean isValidMove = ruleset.isValidMove(WizardCard.ZweiGruen);

    assertTrue(isValidMove);
}

@Test

```

```

    public void testFoolBlueOnGreen2OnGreen3() {
        ruleset.playCard(WizardCard.DreiGruen);
        ruleset.setCurrentPlayer(playerState2);

        ruleset.playCard(WizardCard.ZweiGruen);
        ruleset.setCurrentPlayer(playerState3);

        boolean isValidMove = ruleset.isValidMove(WizardCard.NarrBlau);

        assertTrue(isValidMove);
    }
}

```

---

## 6.2 isValidHeartsMove

---

```

package Ruleset;

import static org.junit.Assert.*;
import static org.junit.Assert.assertEquals;

import org.junit.Before;
import org.junit.Test;

import test.TestGameServer;
import test.TestLobbyServer;
import test.TestPlayer;

import Server.GameServer;
import Server.LobbyServer;
import Server.Player;

public class TestIsValidMoveHearts {
    ServerRuleset ruleset;

    GameServer gameServer;

    LobbyServer lobbyServer;

    Player player;
    String player1;
    String player2;
    String player3;
    String player4;

    PlayerState playerState1;
    PlayerState playerState2;
    PlayerState playerState3;
    PlayerState playerState4;

    @Before
    public void setUp() throws Exception {
        player1 = "Tick";
        player2 = "Trick";
        player3 = "Track";
        player3 = "Duck";
        lobbyServer = new TestLobbyServer();
        player = new TestPlayer(lobbyServer, null, null);
        gameServer = new TestGameServer(lobbyServer, player, "Mein Spiel",
            RulesetType.Hearts, "", false);
    }
}

```

---

```

        ruleset = new ServerHearts(gameServer);

        ruleset.addPlayerToGame(player1);
        ruleset.addPlayerToGame(player2);
        ruleset.addPlayerToGame(player3);
        ruleset.addPlayerToGame(player4);

        playerState1 = ruleset.getPlayerState(player1);
        playerState2 = ruleset.getPlayerState(player2);
        playerState3 = ruleset.getPlayerState(player3);

        ruleset.setFirstPlayer(playerState1);
    }

    @Test
    public void testIsValidMove() {
        ruleset.giveACard(playerState1, HeartsCard.Herz2);
        ruleset.giveACard(playerState1, HeartsCard.Kreuz9);
        ruleset.giveACard(playerState2, HeartsCard.Caro3);
        ruleset.giveACard(playerState2, HeartsCard.Caro6);
        ruleset.giveACard(playerState3, HeartsCard.Pik4);
        ruleset.giveACard(playerState3, HeartsCard.Pik5);
        ruleset.giveACard(playerState4, HeartsCard.Pik1);
        ruleset.giveACard(playerState4, HeartsCard.Herz7);

        boolean isValidMove = ruleset.isValidMove(HeartsCard.Herz2);

        assertFalse(isValidMove);

        boolean isValidMove2 = ruleset.isValidMove(HeartsCard.Caro3);

        assertTrue(isValidMove2);
    }

    @Test
    public void testIsValidMoveOnlyHearts() {
        ruleset.giveACard(playerState1, HeartsCard.Herz2);
        ruleset.giveACard(playerState1, HeartsCard.Herz5);
        ruleset.giveACard(playerState2, HeartsCard.Caro3);
        ruleset.giveACard(playerState2, HeartsCard.Caro6);
        ruleset.giveACard(playerState3, HeartsCard.Pik4);
        ruleset.giveACard(playerState3, HeartsCard.Pik5);
        ruleset.giveACard(playerState4, HeartsCard.Pik1);
        ruleset.giveACard(playerState4, HeartsCard.Herz7);

        boolean isValidMove = ruleset.isValidMove(HeartsCard.Herz2);

        assertTrue(isValidMove);

        boolean isValidMove2 = ruleset.isValidMove(HeartsCard.Herz5);

        assertTrue(isValidMove2);
    }
}

```

## 6.3 getWinner

```

package Ruleset;

import static org.junit.Assert.*;

```

```
import java.util.List;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import testKlassen.TestPlayer;
import ComObjects.ComObject;
import ComObjects.ComRuleset;
import ComObjects.MsgGameEnd;
import Server.GameServer;
import Server.LobbyServer;
import Server.Player;

public class TestHeartsWinner {

    LobbyServer lobbyServer;

    GameServer gameServer;

    ServerRuleset heartsServerRuleset;

    TestPlayer blue;

    TestPlayer white;

    TestPlayer orange;

    TestPlayer brown;

    List<ComObject> inputList;

    ComRuleset comObject;

    MsgGameEnd endMsg;

    String winner;

    @Before
    public void setUp() {
        lobbyServer = new LobbyServer();
        blue = new TestPlayer(lobbyServer, null, null);
        white = new TestPlayer(lobbyServer, null, null);
        orange = new TestPlayer(lobbyServer, null, null);
        brown = new TestPlayer(lobbyServer, null, null);
    }

    @After
    public void tearDown() {
        blue = null;
        white = null;
        orange = null;
        brown = null;
        lobbyServer = null;
        gameServer = null;
        inputList = null;
        comObject = null;
        endMsg = null;
        winner = null;
    }

    @Test
    public void testGetWinner() {
```

```

gameServer = new GameServer(lobbyServer, blue, "Test Game",
    RulesetType.Wizard, "", false);
gameServer.addPlayer(white);
gameServer.addPlayer(orange);
gameServer.addPlayer(brown);

heartsServerRuleset = new ServerWizard(gameServer);
heartsServerRuleset.addPlayerToGame("Mr. Blue");
heartsServerRuleset.addPlayerToGame("Mr. White");
heartsServerRuleset.addPlayerToGame("Mr. Orange");
heartsServerRuleset.addPlayerToGame("Mr. Brown");
heartsServerRuleset.setPoints(heartsServerRuleset.getPlayerState("Mr.
    Blue"), 80);
heartsServerRuleset.setPoints(heartsServerRuleset.getPlayerState("Mr.
    White"), 20);
heartsServerRuleset.setPoints(heartsServerRuleset.getPlayerState("Mr.
    Orange"), 60);
heartsServerRuleset.setPoints(heartsServerRuleset.getPlayerState("Mr.
    Brown"), 110);
heartsServerRuleset.setGamePhase(GamePhase.Ending);
heartsServerRuleset.calculateRoundOutcome();

assertTrue(heartsServerRuleset.getWinner().equals("Mr. Brown"));

inputList = blue.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an Blue", "Mr. Brown", winner);

inputList = white.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an White", "Mr. Brown", winner);

inputList = orange.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an Orange", "Mr. Brown", winner);

inputList = brown.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an Brown", "Mr. Brown", winner);
    }
}

```

---

```

package Ruleset;

import static org.junit.Assert.*;

import java.util.List;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import testKlassen.TestPlayer;
import ComObjects.ComChatMessage;
import ComObjects.ComObject;

```

```
import ComObjects.ComRuleset;
import ComObjects.MsgGameEnd;
import Server.GameServer;
import Server.LobbyServer;

/**
 * Testet ob der richtige Sieger ermittelt wird und ob jedem Mitspieler
 * der richtige Sieger mitgeteilt wird
 */
public class TestWizardWinner {

    LobbyServer lobbyServer;

    GameServer gameServer;

    ServerRuleset wizardServerRuleset;

    TestPlayer blue;

    TestPlayer white;

    TestPlayer orange;

    TestPlayer brown;

    List<ComObject> inputList;

    ComRuleset comObject;

    MsgGameEnd endMsg;

    String winner;

    @Before
    public void setUp() {
        lobbyServer = new LobbyServer();
        blue = new TestPlayer(lobbyServer, null, null);
        white = new TestPlayer(lobbyServer, null, null);
        orange = new TestPlayer(lobbyServer, null, null);
        brown = new TestPlayer(lobbyServer, null, null);
    }

    @After
    public void tearDown() {
        blue = null;
        white = null;
        orange = null;
        brown = null;
        lobbyServer = null;
        gameServer = null;
        inputList = null;
        inputList = null;
        comObject = null;
        endMsg = null;
        winner = null;
    }

    @Test
    public void testGetWinner() {

        gameServer = new GameServer(lobbyServer, blue, "Test Game",
            RulesetType.Wizard, "", false);
        gameServer.addPlayer(white);
```

```

gameServer.addPlayer(orange);
gameServer.addPlayer(brown);

wizardServerRuleset = new ServerWizard(gameServer);
wizardServerRuleset.addPlayerToGame("Mr. Blue");
wizardServerRuleset.addPlayerToGame("Mr. White");
wizardServerRuleset.addPlayerToGame("Mr. Orange");
wizardServerRuleset.addPlayerToGame("Mr. Brown");
wizardServerRuleset.setPoints(wizardServerRuleset.getPlayerState("Mr.
    Blue"),80);
wizardServerRuleset.setPoints(wizardServerRuleset.getPlayerState("Mr.
    White"),200);
wizardServerRuleset.setPoints(wizardServerRuleset.getPlayerState("Mr.
    Orange"),130);
wizardServerRuleset.setPoints(wizardServerRuleset.getPlayerState("Mr.
    Brown"),240);
wizardServerRuleset.setGamePhase(GamePhase.Ending);
wizardServerRuleset.calculateRoundOutcome();

assertTrue(wizardServerRuleset.getWinner().equals("Mr. Brown"));

inputList = blue.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an Blue", "Mr. Brown", winner);

inputList = white.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an White", "Mr. Brown", winner);

inputList = orange.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an Orange", "Mr. Brown", winner);

inputList = brown.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an Brown", "Mr. Brown", winner);
    }
}

```

## 6.4 QuitPlayer

```
ackage Server;
```

```

import static org.junit.Assert.*;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;

```

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import test.TestGameServer;
import test.TestLobbyServer;
import test.TestPlayer;

import ComObjects.*;
import ComObjects.ComWarning;
import Ruleset.RulesetType;

public class QuitGameTest {

    TestLobbyServer lobby;

    TestPlayer player1;

    TestPlayer player2;
    TestPlayer player3;
    TestPlayer player4;

    TestGameServer game;

    ComClientQuit quit;

    @Before
    public void setUp() throws Exception {
        lobby = new TestLobbyServer();

        player1 = new TestPlayer(lobby, null, null);
        player1.setName("MrBlue");
        lobby.addPlayer(player1);

        player2 = new TestPlayer(lobby, null, null);
        player2.setName("MrWhite");
        player3 = new TestPlayer(lobby, null, null);
        player3.setName("MrPink");
        player4 = new TestPlayer(lobby, null, null);
        player4.setName("MrRed");

        game = new TestGameServer(lobby, player1, "MrBluesGame",
            RulesetType.Hearts, null, false);

        game.addPlayer(player2);
        game.addPlayer(player3);
        game.addPlayer(player4);

        quit = new ComClientQuit();
    }

    @After
    public void tearDown() throws Exception {
        lobby = null;
        player1 = null;
        player2 = null;
        player3 = null;
        player4 = null;
        game = null;
    }

    @Test
    public void testPlayerQuitGame() throws IOException{
```



```

player1.changeServer(game);
assertTrue(game.initLobby().getPlayerList().contains(player1.getName()));

ComInitLobby initLobby = lobby.initLobby();
ComWarning warning = new ComWarning("Ein Spieler hat das Spiel
    verlassen");

player1.injectComObject(quit);

assertFalse(lobby.initLobby().getGameList().contains(game));
assertTrue(lobby.initLobby().getPlayerList().contains(player1.getName()));
assertTrue(lobby.initLobby().getPlayerList().contains(player1.getName()));
assertTrue(lobby.initLobby().getPlayerList().contains(player1.getName()));
assertTrue(lobby.initLobby().getPlayerList().contains(player1.getName()));

assertTrue(player1.getServerInput().get(0).getClass() ==
    initLobby.getClass());
assertTrue(player1.getServerInput().get(1).getClass() ==
    warning.getClass());
assertTrue(player2.getServerInput().get(0).getClass() ==
    initLobby.getClass());
assertTrue(player2.getServerInput().get(1).getClass() ==
    warning.getClass());
assertTrue(player3.getServerInput().get(0).getClass() ==
    initLobby.getClass());
assertTrue(player3.getServerInput().get(1).getClass() ==
    warning.getClass());
assertTrue(player4.getServerInput().get(0).getClass() ==
    initLobby.getClass());
assertTrue(player4.getServerInput().get(1).getClass() ==
    warning.getClass());
    }
}

```

---

## 6.5 Chat

---

```

package chat;

import static org.junit.Assert.*;

import org.junit.Test;
import org.junit.After;
import org.junit.Before;

import testKlassen.TestMessageListenerThread;
import testKlassen.TestObserver;
import Client.MessageListenerThread;
import Client.ClientModel;
import ComObjects.ComChatMessage;

public class ClientModelChatTest {

    ComChatMessage testMessage;

    ClientModel testModel;

    TestObserver testObserver;

    TestMessageListenerThread testNetIO;

```

---

```

        String testText;

        @Before
        public void setUp() {
            testNetIO = new TestMessageListenerThread();
            testObserver = new TestObserver();
            testMessage = new ComChatMessage("Hello Test!");
            testModel = new ClientModel((MessageListenerThread) testNetIO);
            testNetIO.setModel(testModel);
            testModel.addObserver(testObserver);
        }

        @After
        public void tearDown() {
            testNetIO = null;
            testMessage = null;
            testModel = null;
            testObserver = null;
        }

        @Test
        public void testSendChatMessage() {
            String inputText = "Hello Test!";
            testModel.sendChatMessage(inputText);
            testText = ((ComChatMessage)
                testNetIO.getModelInput().get(0)).getChatMessage();
            assertEquals("Vergleich der gesendeten Chatnachrichten", testText,
                inputText);
        }

        @Test
        public void testReceiveChatMessage() {
            testNetIO.injectComObject(testMessage);
            assertTrue("Vergleich der empfangenen Chatnachrichten",
                testObserver.getChatMessage().equals(testMessage.getChatMessage()));
        }
    }
}

```

---

```

package chat;

import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import testKlassen.TestPlayer;
import Server.LobbyServer;
import ComObjects.ComChatMessage;

public class LobbyServerChatTest {

    ComChatMessage testMessage;

    LobbyServer testServer;

    TestPlayer player1;

    TestPlayer player2;

    TestPlayer player3;

    String testText1;

```

---

```

String testText2;

String testText3;

@Before
public void setUp() {
    testMessage = new ComChatMessage("Hello Test!");
    testServer = new LobbyServer();
    player1 = new TestPlayer(testServer, null, null);
    player2 = new TestPlayer(testServer, null, null);
    player3 = new TestPlayer(testServer, null, null);
}

@After
public void tearDown() {
    testMessage = null;
    testServer = null;
    player1 = null;
    player2 = null;
    player3 = null;
    testText1 = null;
    testText2 = null;
    testText3 = null;
}

@Test
public void testReceiveMessagePlayerComChatMessage() {
    String messageToMatch = testMessage.getChatMessage();
    testServer.addPlayer(player1);
    testServer.addPlayer(player2);
    testServer.addPlayer(player3);
    player1.injectComObject(testMessage);
    testText1 = ((ComChatMessage)
        player1.getServerInput()).getChatMessage();
    testText2 = ((ComChatMessage)
        player2.getServerInput()).getChatMessage();
    testText3 = ((ComChatMessage)
        player3.getServerInput()).getChatMessage();
    assertEquals("Nachricht an Spieler 1", messageToMatch, testText1);
    assertEquals("Nachricht an Spieler 2", messageToMatch, testText2);
    assertEquals("Nachricht an Spieler 3", messageToMatch, testText3);
}
}

```

## 7 Implementierungsplan

Es werden für jeden Milestone die einzelnen Arbeitspakete angegeben. Die angegebenen Klassen werden nicht sofort vollständig implementiert, sondern mit den vom Arbeitspaket und Milestone verlangten Funktionen ausgestattet.

### 7.1 Milestone 1

Für den ersten Milestone werden folgende Funktionen angestrebt:

Der Nutzer kann sich im Login-Fenster anmelden und die Lobby betreten. Er kann ein Spiel erstellen und offenen Spielen beitreten. Das wird in der Lobby angezeigt, man gelangt jedoch noch nicht ins Wartefenster. Nebenläufig dazu wird die Datenschicht der Regelwerke implementiert.

- View(Login+Lobby) Dauer: 8 Std.  
Klassen: Login, Lobby, Warning, ClientController
- Client(Login) Dauer 8 Std.  
Klassen: ClientMain, ClientModel, MessageListener Thread, ClientState, ViewNotification
- Server(Login) Dauer 16 Std.  
Klassen: Server, ServerMain, LobbyServer, Player, ClientListenerThread, ComObject, ComLoginRequest, ComClientQuit, ComServerAcknowledgement, ComWarning
- Ruleset(Daten) Dauer 20 Std.  
Klassen: Card, Colour, HeartsCard, WizCard, OtherData, WizData, HeartsData, GameClientUpdate, GameState, PlayerState, RulesetType
- Client(Lobby) Dauer 8 Std.  
Klassen: ClientModel
- Server(Lobby) Dauer 8 Std.  
Klassen: LobbyServer, ComChatMessage, ComLobbyUpdateGamelist, ComJoinRequest, ComInitLobby, ComUpdatePlayerlist
- View(Create+Join) Dauer 8 Std.  
Klassen: Password, CreateGame, ClientController
- Client(Create+Join) Dauer 8 Std.  
Klassen: ClientModel
- Server(Create+Join) Dauer 12 Std.  
Klassen: LobbyServer, ComJoinRequest, ComCreateGameRequest

## 7.2 Milestone 2

Für den zweiten Milestone werden folgende Funktionen angestrebt:

Beim Beitreten oder Erstellen eines Spiels gelangt man ins Wartefenster. Der Spielleiter kann hier Spieler entfernen. Diese gelangen zurück in die Lobby. Das Spiel kann noch nicht gestartet werden, aber das Regelwerk wird bereits serverseitig implementiert.

- Ruleset(Wizard-Server) Dauer 30 Std.  
Klassen: ServerRuleset, ServerWizard, RulesetMessage, MsgCard, MsgCardRequest, MsgGameEnd, MsgNumber, MsgNumberRequest, MsgSelection, MsgSelectionRequest, MsgUser
- View(GameLobby) Dauer 8 Std.  
Klassen: GameLobby, ClientController
- Client(GameLobby) Dauer 8 Std.  
Klassen: ClientModel
- Server(GameLobby) Dauer 10 Std.  
Klassen: LobbyServer, GameServer, ComBeenKicked, ComClientLeave, ComInitGameLobby, ComKickPlayerRequest, ComStartGame
- View(Game) Dauer 20 Std.  
Klassen: Game, GamePanel, OtherPlayer, OwnHand, ViewCard, DrawDeck, DiscardPile, ScoreWindow, ClientController
- Client(Game) Dauer 14 Std.  
Klassen: ClientModel

### 7.3 Milestone 3

Für den dritten Milestone werden folgende Funktionen angestrebt:  
Es kann schon eine vollständige Partie Wizard gespielt werden.

- Server(Game) Dauer 4 Std.  
Klassen: GameServer, ComStartGame, ComRuleset, ComGameEnd
- Ruleset(Wizard-Client) Dauer 12 Std.  
Klassen: ClientWizard
- View(WizardWindows) Dauer 4 Std.  
Klassen: Chooseltem, InputNumber, ClientController
- Client(Wizard) Dauer 6 Std.  
Klassen: ClientModel
- View(HeartsWindows) Dauer 2 Std.  
Klassen: ChooseCards, ClientController
- Client(Hearts) Dauer 4 Std.  
Klassen: ClientModel
- Ruleset(Hearts-Server) Dauer 16 Std.  
Klassen: ServerHearts

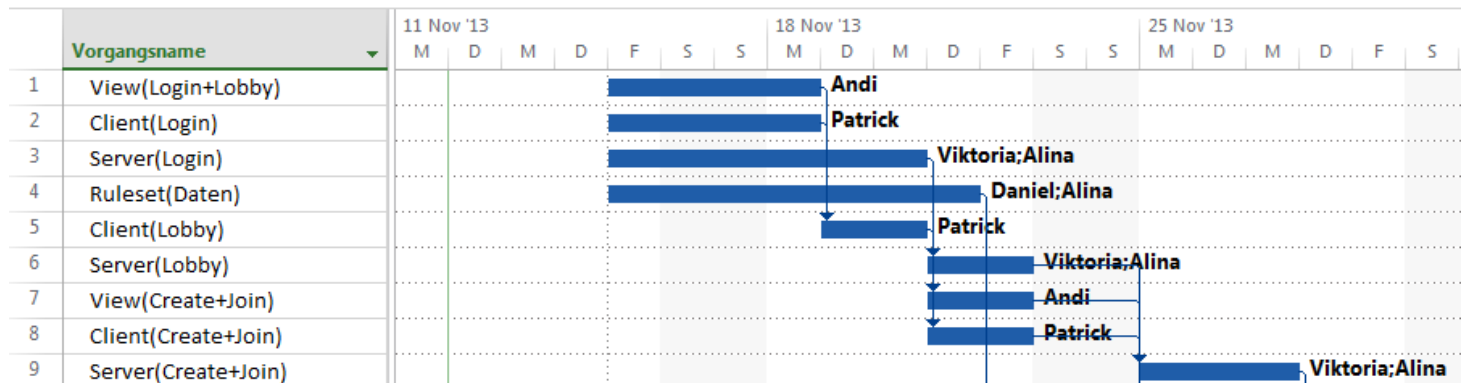
### 7.4 Finale Version

Die finale Version enthält die volle Funktionalität des Programs. Es können also sowohl Wizard als auch Hearts gespielt werden.

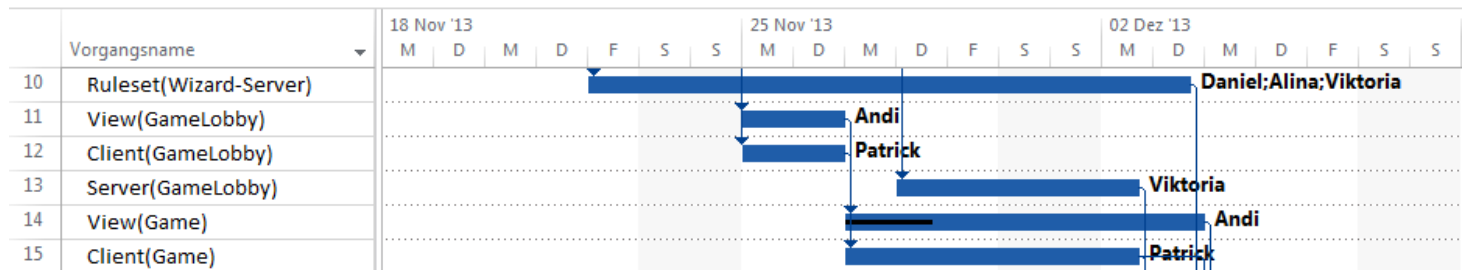
- Ruleset(Hearts-Client) Dauer 10Std.  
Klassen: ClientHearts
- ViewPolishing(evtl Tests) Dauer 10Std  
Verbesserungen an der bisherigen Implementierung. Gegebenfalls Schreiben von zusätzlichen Tests
- ClientPolishing(evtl Tests) Dauer 10Std  
Verbesserungen an der bisherigen Implementierung. Gegebenfalls Schreiben von zusätzlichen Tests
- ServerPolishing(evtl Tests) Dauer 10Std  
Verbesserungen an der bisherigen Implementierung. Gegebenfalls Schreiben von zusätzlichen Tests
- RulesetPolishing(evtl Tests) Dauer 10Std  
Verbesserungen an der bisherigen Implementierung. Gegebenfalls Schreiben von zusätzlichen Tests

## 7.5 Gantt-Diagramme

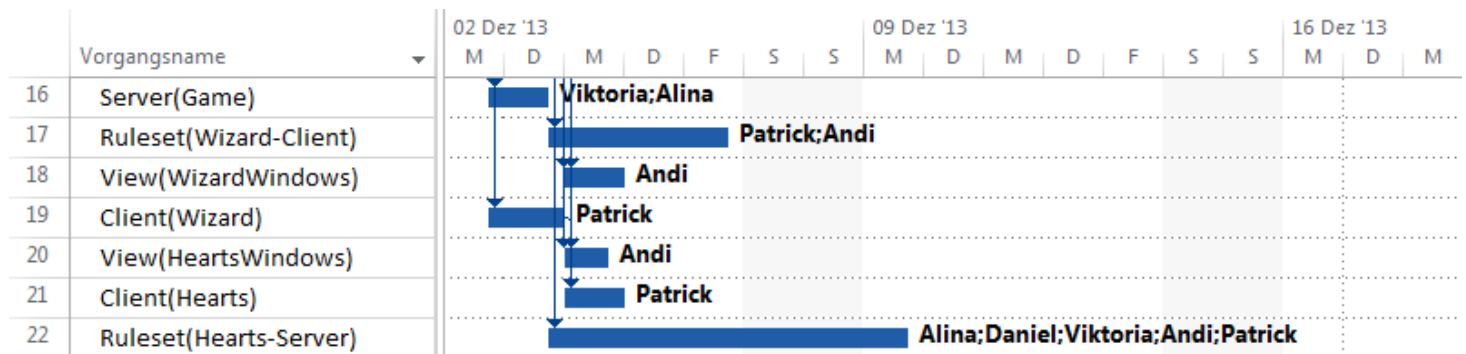
Milestone 1:



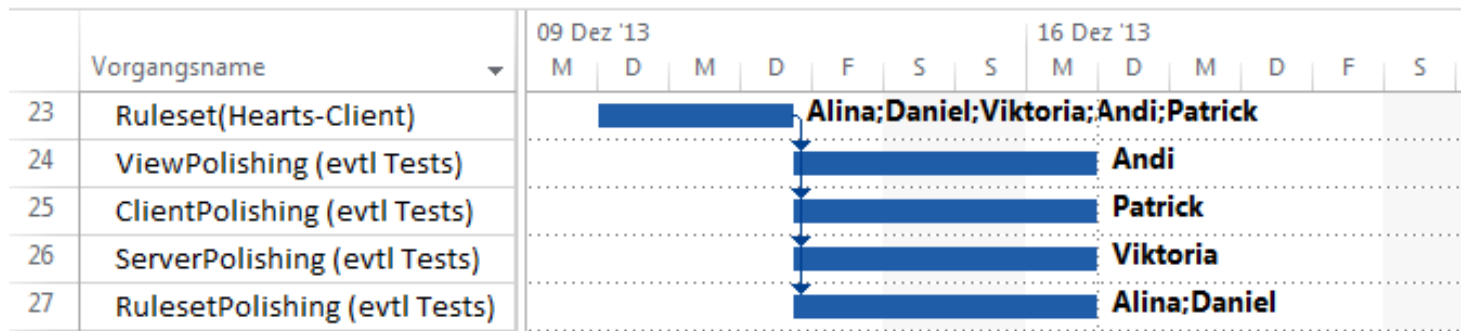
Milestone 2:



Milestone 3:



Finale Version:



## Index

- addCard
  - Ruleset::PlayerState, [87](#)
- addChatMessageListener
  - Client::View::GameLobby, [27](#)
  - Client::View::Lobby, [31](#)
- addConnectButtonListener
  - Client::View::Login, [32](#)
- addCreateButtonListener
  - Client::View::CreateGame, [24](#)
- addGameServer
  - Server::LobbyServer, [106](#)
- addHostButtonListener
  - Client::View::Lobby, [30](#)
- addJoinButtonListener
  - Client::View::Lobby, [30](#)
  - Client::View::Password, [34](#)
- addLanguageSelectionListener
  - Client::View::Login, [32](#)
- addLeaveButtonListener
  - Client::View::CreateGame, [24](#)
  - Client::View::GameLobby, [27](#)
  - Client::View::Lobby, [30](#)
- addName
  - Server::LobbyServer, [106](#)
- addPanelMouseListener
  - Client::View::CreateGame, [23](#)
- addPlayer
  - Server::GameServer, [102](#)
  - Server::Server, [111](#)
- addPlayerToGame
  - Ruleset::GameState, [81](#)
  - Ruleset::ServerRuleset, [93](#)
- addRemoveButtonListener
  - Client::View::GameLobby, [27](#)
- addRulesetSelectionListener
  - Client::View::CreateGame, [24](#)
- addStartButtonListener
  - Client::View::GameLobby, [27](#)
- announceWinner
  - Client::ClientModel, [19](#)
  - Ruleset::ClientRuleset, [75](#)
- areValidChosenCards
  - Ruleset::ClientHearts, [72](#)
  - Ruleset::ServerHearts, [89](#)
- broadcast
  - Ruleset::ServerRuleset, [94](#)
  - Server::Server, [111](#)
- broadcastRulesetMessage
  - Server::GameServer, [102](#)
- changeServer
  - Server::Player, [109](#)
- Client.ClientController, [11](#)
- Client.ClientMain, [12](#)
- Client.ClientModel, [12](#)
- Client.ClientState, [20](#)
- Client.MessageListenerThread, [21](#)
- Client.View.ChooseCards, [21](#)
- Client.View.ChooseItem, [22](#)
- Client.View.CreateGame, [23](#)
- Client.View.DiscardPile, [24](#)
- Client.View.DrawDeck, [24](#)
- Client.View.Game, [24](#)
- Client.View.GameLobby, [26](#)
- Client.View.GamePanel, [28](#)
- Client.View.InputNumber, [28](#)
- Client.View.Language, [29](#)
- Client.View.Lobby, [29](#)
- Client.View.Login, [31](#)
- Client.View.OtherPlayer, [33](#)
- Client.View.OwnHand, [33](#)
- Client.View.Password, [33](#)
- Client.View.ScoreWindow, [34](#)
- Client.View.ViewCard, [35](#)
- Client.View.Warning, [36](#)
- Client.ViewNotification, [37](#)
- Client::ClientMain
  - main, [12](#)
- Client::ClientModel
  - announceWinner, [19](#)
  - ClientModel, [13](#)
  - createConnection, [20](#)
  - getChooseCards, [17](#)
  - getChooseItems, [18](#)
  - getLanguage, [16](#)
  - getLobbyGamelist, [15](#)
  - getOtherPlayerData, [16](#)
  - getOwnHand, [16](#)
  - getOwnScore, [16](#)
  - getPlayedCards, [16](#)
  - getPlayerCount, [17](#)
  - getPlayerlist, [15](#)
  - getRulesets, [20](#)
  - getWarningText, [20](#)
  - getWindowText, [17](#)
  - getWinner, [20](#)
  - giveChosenCards, [17](#)
  - giveChosenItem, [18](#)
  - giveInputNumber, [18](#)
  - hostGame, [17](#)
  - informView, [20](#)
  - initGame, [19](#)
  - joinGame, [19](#)
  - kickPlayer, [16](#)
  - leaveWindow, [14](#)
  - makeMove, [19](#)
  - openChooseCards, [18](#)
  - openChooseItem, [18](#)
  - openInputNumber, [19](#)
  - receiveMessage, [14, 15](#)



- send, 17
  - sendChatMessage, 19
  - setLanguage, 16
  - startGame, 19
- Client::MessageListenerThread
  - startConnection, 21
- Client::View::ChooseCards
  - update, 22
- Client::View::ChooseItem
  - update, 22
- Client::View::CreateGame
  - addCreateButtonListener, 24
  - addLeaveButtonListener, 24
  - addPanelMouseListener, 23
  - addRulesetSelectionListener, 24
  - setLanguage, 24
- Client::View::Game
  - Game, 25
  - makeTrickGameBoard, 25
  - update, 25, 26
- Client::View::GameLobby
  - addChatMessageListener, 27
  - addLeaveButtonListener, 27
  - addRemoveButtonListener, 27
  - addStartButtonListener, 27
  - setLanguage, 27
  - update, 27, 28
- Client::View::InputNumber
  - update, 29
- Client::View::Lobby
  - addChatMessageListener, 31
  - addHostButtonListener, 30
  - addJoinButtonListener, 30
  - addLeaveButtonListener, 30
  - setLanguage, 31
  - update, 31
- Client::View::Login
  - addConnectButtonListener, 32
  - addLanguageSelectionListener, 32
  - setLanguage, 32
  - update, 32
- Client::View::Password
  - addJoinButtonListener, 34
  - setLanguage, 34
  - update, 34
- Client::View::ScoreWindow
  - update, 35
- Client::View::ViewCard
  - getID, 36
  - ViewCard, 35
- Client::View::Warning
  - setText, 36
  - update, 36
- ClientHearts
  - Ruleset::ClientHearts, 71
- ClientModel
  - Client::ClientModel, 13
- ClientRuleset
  - Ruleset::ClientRuleset, 73
- ClientWizard
  - Ruleset::ClientWizard, 76
- ComBeenKicked
  - ComObjects::ComBeenKicked, 38
- ComChatMessage
  - ComObjects::ComChatMessage, 39
- ComCreateGameRequest
  - ComObjects::ComCreateGameRequest, 41
- ComInitGameLobby
  - ComObjects::ComInitGameLobby, 43
- ComInitLobby
  - ComObjects::ComInitLobby, 44
- ComJoinRequest
  - ComObjects::ComJoinRequest, 46
- ComKickPlayerRequest
  - ComObjects::ComKickPlayerRequest, 47
- ComLobbyUpdateGamelist
  - ComObjects::ComLobbyUpdateGamelist, 49
- ComLoginRequest
  - ComObjects::ComLoginRequest, 51
- ComObjects.ComBeenKicked, 37
- ComObjects.ComChatMessage, 38
- ComObjects.ComClientLeave, 39
- ComObjects.ComClientQuit, 40
- ComObjects.ComCreateGameRequest, 41
- ComObjects.ComInitGameLobby, 43
- ComObjects.ComInitLobby, 44
- ComObjects.ComJoinRequest, 45
- ComObjects.ComKickPlayerRequest, 47
- ComObjects.ComLobbyUpdateGamelist, 49
- ComObjects.ComLoginRequest, 50
- ComObjects.ComObject, 51
- ComObjects.ComRuleset, 52
- ComObjects.ComServerAcknowledgement, 53
- ComObjects.ComStartGame, 54
- ComObjects.ComUpdatePlayerlist, 55
- ComObjects.ComWarning, 56
- ComObjects.MsgCard, 57
- ComObjects.MsgCardRequest, 58
- ComObjects.MsgGameEnd, 59
- ComObjects.MsgMultiCards, 61
- ComObjects.MsgMultiCardsRequest, 62
- ComObjects.MsgNumber, 63
- ComObjects.MsgNumberRequest, 65
- ComObjects.MsgSelection, 65
- ComObjects.MsgSelectionRequest, 66
- ComObjects.MsgUser, 67
- ComObjects.RulesetMessage, 68
- ComObjects::ComBeenKicked
  - ComBeenKicked, 38
  - getMessage, 38
  - process, 38
- ComObjects::ComChatMessage
  - ComChatMessage, 39
  - getChatMessage, 39
  - process, 39
- ComObjects::ComClientLeave

- process, 40
- ComObjects::ComClientQuit
  - process, 40, 41
- ComObjects::ComCreateGameRequest
  - ComCreateGameRequest, 41
  - getGameName, 42
  - getPassword, 42
  - getRuleset, 42
  - hasPassword, 42
  - process, 42
- ComObjects::ComInitGameLobby
  - ComInitGameLobby, 43
  - getPlayerList, 43
  - process, 43, 44
- ComObjects::ComInitLobby
  - ComInitLobby, 44
  - getGameList, 45
  - getPlayerList, 45
  - process, 45
- ComObjects::ComJoinRequest
  - ComJoinRequest, 46
  - gameMasterName, 46
  - getGameMasterName, 46
  - process, 46
- ComObjects::ComKickPlayerRequest
  - ComKickPlayerRequest, 47
  - getPlayerName, 47
  - process, 47, 49
- ComObjects::ComLobbyUpdateGamelist
  - ComLobbyUpdateGamelist, 49
  - getGameServer, 50
  - isRemoveFlag, 50
  - process, 50
- ComObjects::ComLoginRequest
  - ComLoginRequest, 51
  - getPlayerName, 51
  - process, 51
- ComObjects::ComObject
  - process, 52
- ComObjects::ComRuleset
  - ComRuleset, 53
  - getRulesetMessage, 53
  - process, 53
- ComObjects::ComServerAcknowledgement
  - process, 54
- ComObjects::ComStartGame
  - process, 54, 55
- ComObjects::ComUpdatePlayerlist
  - ComUpdatePlayerlist, 55
  - getPlayerName, 55
  - isRemoveFlag, 56
  - process, 56
- ComObjects::ComWarning
  - ComWarning, 57
  - getWarning, 57
  - process, 57
- ComObjects::MsgCard
  - getCard, 58
- MsgCard, 58
  - visit, 58
- ComObjects::MsgCardRequest
  - visit, 59
- ComObjects::MsgGameEnd
  - getWinnerName, 61
  - MsgGameEnd, 60
  - visit, 61
- ComObjects::MsgMultiCards
  - getCardList, 62
  - MsgMultiCards, 62
  - visit, 62
- ComObjects::MsgMultiCardsRequest
  - getCount, 63
  - MsgMultiCardsRequest, 63
  - visit, 63
- ComObjects::MsgNumber
  - getNumber, 64
  - MsgNumber, 64
  - visit, 64
- ComObjects::MsgNumberRequest
  - visit, 65
- ComObjects::MsgSelection
  - getSelection, 66
  - MsgSelection, 66
  - visit, 66
- ComObjects::MsgSelectionRequest
  - visit, 67
- ComObjects::MsgUser
  - getGameClientUpdate, 68
  - MsgUser, 67
  - visit, 68
- ComObjects::RulesetMessage
  - visit, 69, 70
- ComRuleset
  - ComObjects::ComRuleset, 53
- ComUpdatePlayerlist
  - ComObjects::ComUpdatePlayerlist, 55
- ComWarning
  - ComObjects::ComWarning, 57
- createConnection
  - Client::ClientModel, 20
- createDeck
  - Ruleset::ServerRuleset, 92
- dealCards
  - Ruleset::GameState, 83
  - Ruleset::ServerRuleset, 95
- Game
  - Client::View::Game, 25
- GameClientUpdate
  - Ruleset::GameClientUpdate, 78
- gameMasterName
  - ComObjects::ComJoinRequest, 46
- GameServer
  - Server::GameServer, 101
- GameServerRepresentation
  - Server::GameServerRepresentation, 105

GameState  
    Ruleset::GameState, 80  
generateGameClientUpdate  
    Ruleset::ServerHearts, 89  
    Ruleset::ServerRuleset, 96  
    Ruleset::ServerWizard, 98  
getAnnouncedTricks  
    Ruleset::WizData, 100  
getCard  
    ComObjects::MsgCard, 58  
getCardList  
    ComObjects::MsgMultiCards, 62  
getCardsLeftInDeck  
    Ruleset::GameState, 81  
getChatMessage  
    ComObjects::ComChatMessage, 39  
getChooseCards  
    Client::ClientModel, 17  
getChooseItems  
    Client::ClientModel, 18  
getColour  
    Ruleset::Card, 70  
    Ruleset::HeartsCard, 84  
    Ruleset::WizardCard, 99  
getCount  
    ComObjects::MsgMultiCardsRequest, 63  
getCurrentPlayer  
    Ruleset::ClientRuleset, 74  
    Ruleset::GameClientUpdate, 79  
    Ruleset::GameState, 81  
    Ruleset::ServerRuleset, 93  
getEndingPoints  
    Ruleset::ServerHearts, 89  
getFirstPlayer  
    Ruleset::GameState, 81  
    Ruleset::ServerRuleset, 92  
getGameClientUpdate  
    ComObjects::MsgUser, 68  
getGameList  
    ComObjects::ComInitLobby, 45  
getGameMasterName  
    ComObjects::ComJoinRequest, 46  
getGameName  
    ComObjects::ComCreateGameRequest, 42  
getGamePhase  
    Ruleset::ClientRuleset, 73  
    Ruleset::ServerRuleset, 92  
getGameServer  
    ComObjects::ComLobbyUpdateGamelist, 50  
getHand  
    Ruleset::PlayerState, 87  
getID  
    Client::View::ViewCard, 36  
getLanguage  
    Client::ClientModel, 16  
getLobbyGamelist  
    Client::ClientModel, 15  
getMaxPlayers  
    Ruleset::ClientRuleset, 73  
    Ruleset::ServerRuleset, 92  
getMessage  
    ComObjects::ComBeenKicked, 38  
getMinPlayers  
    Ruleset::ClientRuleset, 73  
    Ruleset::ServerRuleset, 91  
getName  
    Ruleset::PlayerState, 87  
    Server::Player, 109  
getNumber  
    ComObjects::MsgNumber, 64  
getNumberOfPlayedCards  
    Ruleset::GameState, 82  
getNumberOfTricks  
    Ruleset::OtherData, 86  
getOtherData  
    Ruleset::PlayerState, 87  
    Ruleset::ServerRuleset, 93  
getOtherPlayerData  
    Client::ClientModel, 16  
    Ruleset::ClientRuleset, 74  
    Ruleset::GameClientUpdate, 79  
getOwnData  
    Ruleset::ClientRuleset, 74  
    Ruleset::GameClientUpdate, 79  
getOwnHand  
    Client::ClientModel, 16  
    Ruleset::ClientRuleset, 74  
    Ruleset::GameClientUpdate, 78  
getOwnScore  
    Client::ClientModel, 16  
getPassword  
    ComObjects::ComCreateGameRequest, 42  
getPlayedCards  
    Client::ClientModel, 16  
    Ruleset::GameClientUpdate, 78  
    Ruleset::GameState, 82  
getPlayerCards  
    Ruleset::GameState, 83  
    Ruleset::ServerRuleset, 94  
getPlayerCount  
    Client::ClientModel, 17  
getPlayerList  
    ComObjects::ComInitGameLobby, 43  
    ComObjects::ComInitLobby, 45  
getPlayerName  
    ComObjects::ComKickPlayerRequest, 47  
    ComObjects::ComLoginRequest, 51  
    ComObjects::ComUpdatePlayerlist, 55  
getPlayerState  
    Ruleset::GameState, 82  
    Ruleset::ServerRuleset, 93  
getPlayerlist  
    Client::ClientModel, 15  
getPoints  
    Ruleset::OtherData, 86  
    Ruleset::ServerRuleset, 95

- getRepresentation
  - Server::GameServer, 102
- getRoundNumber
  - Ruleset::GameState, 82
  - Ruleset::ServerRuleset, 91
- getRuleset
  - ComObjects::ComCreateGameRequest, 42
- getRulesetMessage
  - ComObjects::ComRuleset, 53
- getRulesetType
  - Ruleset::ClientRuleset, 73
  - Ruleset::ServerRuleset, 91
- getRulesets
  - Client::ClientModel, 20
- getSelection
  - ComObjects::MsgSelection, 66
- getTrumpCard
  - Ruleset::ClientRuleset, 74
  - Ruleset::GameClientUpdate, 79
  - Ruleset::GameState, 82
- getValue
  - Ruleset::Card, 70
  - Ruleset::HeartsCard, 84
  - Ruleset::WizardCard, 99
- getWarning
  - ComObjects::ComWarning, 57
- getWarningText
  - Client::ClientModel, 20
- getWindowText
  - Client::ClientModel, 17
- getWinner
  - Client::ClientModel, 20
- getWinnerName
  - ComObjects::MsgGameEnd, 61
- getplayingRounds
  - Ruleset::ServerWizard, 97
- giveACard
  - Ruleset::GameState, 83
  - Ruleset::ServerRuleset, 95
- giveChosenCards
  - Client::ClientModel, 17
- giveChosenItem
  - Client::ClientModel, 18
- giveInputNumber
  - Client::ClientModel, 18
- handleIOException
  - Server::GameServer, 104
  - Server::LobbyServer, 108
  - Server::Server, 111
- hasPassword
  - ComObjects::ComCreateGameRequest, 42
- HeartsCard
  - Ruleset::HeartsCard, 84
- hostGame
  - Client::ClientModel, 17
- informView
  - Client::ClientModel, 20
- initGame
  - Client::ClientModel, 19
- initLobby
  - Server::GameServer, 104
  - Server::LobbyServer, 107
- isRemoveFlag
  - ComObjects::ComLobbyUpdateGamelist, 50
  - ComObjects::ComUpdatePlayerlist, 56
- isValidColour
  - Ruleset::ClientWizard, 77
  - Ruleset::ServerWizard, 98
- isValidMove
  - Ruleset::ClientHearts, 71
  - Ruleset::ClientRuleset, 75
  - Ruleset::ClientWizard, 76
  - Ruleset::ServerHearts, 89
  - Ruleset::ServerRuleset, 96
  - Ruleset::ServerWizard, 97
- isValidNumber
  - Ruleset::ServerWizard, 98
- isValidTrickNumber
  - Ruleset::ClientWizard, 77
- joinGame
  - Client::ClientModel, 19
- kickPlayer
  - Client::ClientModel, 16
- leaveWindow
  - Client::ClientModel, 14
- madeTrick
  - Ruleset::OtherData, 86
- main
  - Client::ClientMain, 12
  - Server::ServerMain, 112
- makeMove
  - Client::ClientModel, 19
- makeTrickGameBoard
  - Client::View::Game, 25
- MsgCard
  - ComObjects::MsgCard, 58
- MsgGameEnd
  - ComObjects::MsgGameEnd, 60
- MsgMultiCards
  - ComObjects::MsgMultiCards, 62
- MsgMultiCardsRequest
  - ComObjects::MsgMultiCardsRequest, 63
- MsgNumber
  - ComObjects::MsgNumber, 64
- MsgSelection
  - ComObjects::MsgSelection, 66
- MsgUser
  - ComObjects::MsgUser, 67
- nextPlayer
  - Ruleset::ServerRuleset, 93
- openChooseCards

- Client::ClientModel, 18
- openChooseItem
  - Client::ClientModel, 18
- openInputNumber
  - Client::ClientModel, 19
- password
  - Server::GameServer, 104
- playCard
  - Ruleset::GameState, 83
  - Ruleset::ServerRuleset, 95
- Player
  - Server::Player, 109
- PlayerState
  - Ruleset::PlayerState, 87
- playingRounds
  - Ruleset::ServerWizard, 98
- process
  - ComObjects::ComBeenKicked, 38
  - ComObjects::ComChatMessage, 39
  - ComObjects::ComClientLeave, 40
  - ComObjects::ComClientQuit, 40, 41
  - ComObjects::ComCreateGameRequest, 42
  - ComObjects::ComInitGameLobby, 43, 44
  - ComObjects::ComInitLobby, 45
  - ComObjects::ComJoinRequest, 46
  - ComObjects::ComKickPlayerRequest, 47, 49
  - ComObjects::ComLobbyUpdateGamelist, 50
  - ComObjects::ComLoginRequest, 51
  - ComObjects::ComObject, 52
  - ComObjects::ComRuleset, 53
  - ComObjects::ComServerAcknowledgement, 54
  - ComObjects::ComStartGame, 54, 55
  - ComObjects::ComUpdatePlayerlist, 56
  - ComObjects::ComWarning, 57
- receiveMessage
  - Client::ClientModel, 14, 15
  - Server::GameServer, 103, 104
  - Server::LobbyServer, 106, 107
  - Server::Server, 110
- removeCard
  - Ruleset::PlayerState, 88
- removeGameServer
  - Server::LobbyServer, 106
- removeName
  - Server::LobbyServer, 106
- removePlayer
  - Server::GameServer, 102
  - Server::Server, 111
- removeTricks
  - Ruleset::OtherData, 86
- resolveMessage
  - Ruleset::ClientHearts, 71
  - Ruleset::ClientRuleset, 74, 75
  - Ruleset::ClientWizard, 76, 77
  - Ruleset::ServerHearts, 89
  - Ruleset::ServerRuleset, 94
  - Ruleset::ServerWizard, 97, 98
  - Ruleset.Card, 70
  - Ruleset.ClientHearts, 71
  - Ruleset.ClientRuleset, 72
  - Ruleset.ClientWizard, 76
  - Ruleset.Colour, 77
  - Ruleset.GameClientUpdate, 77
  - Ruleset.GamePhase, 79
  - Ruleset.GameState, 80
  - Ruleset.HeartsCard, 84
  - Ruleset.HeartsData, 85
  - Ruleset.OtherData, 85
  - Ruleset.PlayerState, 86
  - Ruleset.RulesetType, 88
  - Ruleset.ServerHearts, 88
  - Ruleset.ServerRuleset, 90
  - Ruleset.ServerWizard, 96
  - Ruleset.WizData, 100
  - Ruleset.WizardCard, 99
  - Ruleset::Card
    - getColour, 70
    - getValue, 70
  - Ruleset::ClientHearts
    - areValidChosenCards, 72
    - ClientHearts, 71
    - isValidMove, 71
    - resolveMessage, 71
  - Ruleset::ClientRuleset
    - announceWinner, 75
    - ClientRuleset, 73
    - getCurrentPlayer, 74
    - getGamePhase, 73
    - getMaxPlayers, 73
    - getMinPlayers, 73
    - getOtherPlayerData, 74
    - getOwnData, 74
    - getOwnHand, 74
    - getRulesetType, 73
    - getTrumpCard, 74
    - isValidMove, 75
    - resolveMessage, 74, 75
    - send, 75
  - Ruleset::ClientWizard
    - ClientWizard, 76
    - isValidColour, 77
    - isValidMove, 76
    - isValidTrickNumber, 77
    - resolveMessage, 76, 77
  - Ruleset::GameClientUpdate
    - GameClientUpdate, 78
    - getCurrentPlayer, 79
    - getOtherPlayerData, 79
    - getOwnData, 79
    - getOwnHand, 78
    - getPlayedCards, 78
    - getTrumpCard, 79
  - Ruleset::GameState
    - addPlayerToGame, 81
    - dealCards, 83

- GameState, 80
- getCardsLeftInDeck, 81
- getCurrentPlayer, 81
- getFirstPlayer, 81
- getNumberOfPlayedCards, 82
- getPlayedCards, 82
- getPlayerCards, 83
- getPlayerState, 82
- getRoundNumber, 82
- getTrumpCard, 82
- giveACard, 83
- playCard, 83
- setCurrentPlayer, 81
- setFirstPlayer, 81
- setTrumpCard, 82
- Ruleset::HeartsCard
  - getColour, 84
  - getValue, 84
  - HeartsCard, 84
- Ruleset::OtherData
  - getNumberOfTricks, 86
  - getPoints, 86
  - madeTrick, 86
  - removeTricks, 86
  - setPoints, 86
- Ruleset::PlayerState
  - addCard, 87
  - getHand, 87
  - getName, 87
  - getOtherData, 87
  - PlayerState, 87
  - removeCard, 88
- Ruleset::ServerHearts
  - areValidChosenCards, 89
  - generateGameClientUpdate, 89
  - getEndingPoints, 89
  - isValidMove, 89
  - resolveMessage, 89
- Ruleset::ServerRuleset
  - addPlayerToGame, 93
  - broadcast, 94
  - createDeck, 92
  - dealCards, 95
  - generateGameClientUpdate, 96
  - getCurrentPlayer, 93
  - getFirstPlayer, 92
  - getGamePhase, 92
  - getMaxPlayers, 92
  - getMinPlayers, 91
  - getOtherData, 93
  - getPlayerCards, 94
  - getPlayerState, 93
  - getPoints, 95
  - getRoundNumber, 91
  - getRulesetType, 91
  - giveACard, 95
  - isValidMove, 96
  - nextPlayer, 93
  - playCard, 95
  - resolveMessage, 94
  - send, 94
  - ServerRuleset, 91
  - setCurrentPlayer, 93
  - setFirstPlayer, 92
  - setGamePhase, 92
  - setPoints, 94
  - setTrumpCard, 95
- Ruleset::ServerWizard
  - generateGameClientUpdate, 98
  - getplayingRounds, 97
  - isValidColour, 98
  - isValidMove, 97
  - isValidNumber, 98
  - playingRounds, 98
  - resolveMessage, 97, 98
  - setPlayingRounds, 97
- Ruleset::WizData
  - getAnnouncedTricks, 100
  - setAnnouncedTricks, 100
- Ruleset::WizardCard
  - getColour, 99
  - getValue, 99
  - WizardCard, 99
- run
  - Server::Player, 109
- send
  - Client::ClientModel, 17
  - Ruleset::ClientRuleset, 75
  - Ruleset::ServerRuleset, 94
  - Server::Player, 109
- sendChatMessage
  - Client::ClientModel, 19
- sendRulesetMessage
  - Server::GameServer, 102
- sendToPlayer
  - Server::Server, 110
- Server.GameServer, 100
- Server.GameServerRepresentation, 104
- Server.LobbyServer, 105
- Server.LobbyServer.ClientListenerThread, 108
- Server.Player, 108
- Server.Server, 110
- Server.ServerMain, 111
- Server::GameServer
  - addPlayer, 102
  - broadcastRulesetMessage, 102
  - GameServer, 101
  - getRepresentation, 102
  - handleIOException, 104
  - initLobby, 104
  - password, 104
  - receiveMessage, 103, 104
  - removePlayer, 102
  - sendRulesetMessage, 102
- Server::GameServerRepresentation
  - GameServerRepresentation, 105

Server::LobbyServer  
  addGameServer, 106  
  addName, 106  
  handleIOException, 108  
  initLobby, 107  
  receiveMessage, 106, 107  
  removeGameServer, 106  
  removeName, 106  
Server::Player  
  changeServer, 109  
  getName, 109  
  Player, 109  
  run, 109  
  send, 109  
  setName, 110  
Server::Server  
  addPlayer, 111  
  broadcast, 111  
  handleIOException, 111  
  receiveMessage, 110  
  removePlayer, 111  
  sendToPlayer, 110  
Server::ServerMain  
  main, 112  
ServerRuleset  
  Ruleset::ServerRuleset, 91  
setAnnouncedTricks  
  Ruleset::WizData, 100  
setCurrentPlayer  
  Ruleset::GameState, 81  
  Ruleset::ServerRuleset, 93  
setFirstPlayer  
  Ruleset::GameState, 81  
  Ruleset::ServerRuleset, 92  
setGamePhase  
  Ruleset::ServerRuleset, 92  
setLanguage  
  Client::ClientModel, 16  
  Client::View::CreateGame, 24  
  Client::View::GameLobby, 27  
  Client::View::Lobby, 31  
  Client::View::Login, 32  
  Client::View::Password, 34  
setName  
  Server::Player, 110  
setPlayingRounds  
  Ruleset::ServerWizard, 97  
setPoints  
  Ruleset::OtherData, 86  
  Ruleset::ServerRuleset, 94  
setText  
  Client::View::Warning, 36  
setTrumpCard  
  Ruleset::GameState, 82  
  Ruleset::ServerRuleset, 95  
startConnection  
  Client::MessageListenerThread, 21  
startGame  
  Client::ClientModel, 19  
update  
  Client::View::ChooseCards, 22  
  Client::View::ChooseItem, 22  
  Client::View::Game, 25, 26  
  Client::View::GameLobby, 27, 28  
  Client::View::InputNumber, 29  
  Client::View::Lobby, 31  
  Client::View::Login, 32  
  Client::View::Password, 34  
  Client::View::ScoreWindow, 35  
  Client::View::Warning, 36  
ViewCard  
  Client::View::ViewCard, 35  
visit  
  ComObjects::MsgCard, 58  
  ComObjects::MsgCardRequest, 59  
  ComObjects::MsgGameEnd, 61  
  ComObjects::MsgMultiCards, 62  
  ComObjects::MsgMultiCardsRequest, 63  
  ComObjects::MsgNumber, 64  
  ComObjects::MsgNumberRequest, 65  
  ComObjects::MsgSelection, 66  
  ComObjects::MsgSelectionRequest, 67  
  ComObjects::MsgUser, 68  
  ComObjects::RulesetMessage, 69, 70  
WizardCard  
  Ruleset::WizardCard, 99