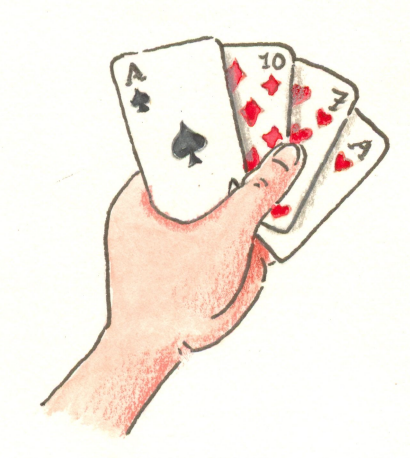
SPEZIFIKATION

10. 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

1 Hierarchie-Verzeichnis

1.1 Klassenhierarchie

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

2 Klassen-Verzeichnis

2.1 Auflistung der Klassen

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

3 Klassen-Dokumentation

3.1 Client.CardID Enum-Referenz

3.2 Client.ClientController Klassenreferenz

3.3 Client.ClientMain Klassenreferenz

Öffentliche, statische Methoden

- static void main (final String[] args)

3.3.1 Dokumentation der Elementfunktionen

3.3.1.1 static void Client.ClientMain.main (final String[] *args*) [static]

Parameter

args	

3.4 Client.ClientModel Klassenreferenz

Abgeleitet von Observable.

Klassen

- class **MessageListenerThread**

Öffentliche Methoden

- void receiveMessage (ComServerAcknowledgement ack)
- List< String > getPlayerlist ()
- Set< GameServerRepresentation > getLobbyGamelist ()
- String getChatMessage ()
- Card getPlayedCard ()
- Card[] getOwnHand ()
- List< OtherData > getOtherPlayerData ()
- int getOwnScore ()
- void setLanguage (final Language language)
- Language getLanguage ()
- void kickPlayer (final String name)
- void hostGame (String gameName, String password)
- void joinGame (final String name, final String password)
- void makeMove (Card card)
- void createConnection (final String username, final String serverAdress, final int port)
- String getWarningText ()
- RulesetType[] getRulesets ()

3.4.1 Ausführliche Beschreibung

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

3.4.2 Dokumentation der Elementfunktionen

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

Diese Hilfsmethode wird von receiveMessage() aufgerufen, falls ein Server Acknowledgement auftritt.

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

Parameter

ack	Eine Bestätigung durch den Server.

3.4.2.2 List<String> Client.ClientModel.getPlayerlist ( )

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

Rückgabe

List Eine Liste der Spieler in der Spiellobby.

3.4.2.3 Set<GameServerRepresentation> Client.ClientModel.getLobbyGamelist ( )

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

Rückgabe

Set Enthält alle Spiele in der ServerLobby.

3.4.2.4 String Client.ClientModel.getChatMessage ( )

Diese Methode wird von der View aufgerufen um eine neue Chatnachricht abzuholen.

Rückgabe

String die Chatnachricht.

3.4.2.5 Card Client.ClientModel.getPlayedCard ( )

Gibt der View die gespielte Karte eines anderen Spielers zurück.

Rückgabe

enum CardID. Die Id der Karte

3.4.2.6 Card [] Client.ClientModel.getOwnHand ( )

Gibt der View die eigenen Spielkarten zurück.

Parameter

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

3.4.2.7 List<OtherData> Client.ClientModel.getOtherPlayerData ( )

Liefert zusätzliche Daten anderer Spieler zurück.

Rückgabe

List<OtherData> Liste mit gespielten Karten.

3.4.2.8 int Client.ClientModel.getOwnScore ( )

Gibt den Punktestand des Spielers aus.

Rückgabe

int Der eigene Punktestand.

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

Setzt die Sprache der GUI.

Parameter

language	Enumerator der die Spielsprache anzeigt.

3.4.2.10 Language Client.ClientModel.getLanguage ( )

Liefert die Sprache der GUI.

Rückgabe

language Enumerator der die Spielsprache anzeigt.

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

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

Parameter

name	des Spielers welcher enfernt werden soll.

3.4.2.12 void Client.ClientModel.hostGame (String *gameName*, String *password*)

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

Parameter

gameName	String Name des Spieles.
password	String Passwort zum sichern des Spieles.

3.4.2.13 void Client.ClientModel.joinGame (final String *name*, final String *password*)

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

Parameter

name	String Der Name des Spiels.
password	String Passwort eines Spieles.

3.4.2.14 void Client.ClientModel.makeMove (Card *card*)

Wird vom ClientConroller aufgerufen um eine Karte auszuspielen.

Parameter

id	Die id der gespielten Karte um sie einer logischen Karte zuordnen zu können.

3.4.2.15 void Client.ClientModel.createConnection (final String *username*, final String *serverAdress*, final int *port*)

Erstellt den MessageListenerThread und führt den Benutzerlogin durch.

Parameter

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

3.4.2.16 String Client.ClientModel.getWarningText ( )

Gibt den Text aus der bei einer Spielwarnung angezeigt wird.

Rückgabe

String Text der Warnung.

3.4.2.17 RulesetType [] Client.ClientModel.getRulesets ( )

Liefert ein Array mit allen implementierten Regelwerken.

Parameter

RulesetType[]	Array von unterstützten Regelwerken.

3.5 Client.ClientModelChatTest Klassenreferenz

3.6 Client.ClientState Enum-Referenz

3.7 Client.LoginError Enum-Referenz

3.8 Client.MVMessages Schnittstellenreferenz

3.9 Client.View.Card Klassenreferenz

Abgeleitet von JPanel.

Öffentliche Methoden

- Card (String s, int n)
- int getID ()

3.9.1 Ausführliche Beschreibung

Sie wird verwendet um einzelne Karten auf das Spielfeld zu zeichnen. Dazu enthï¿½lt sie die Pfadangabe zu dem Ordner, in dem die Bilder der Karten gespeichert sind, und eine ID, um das genaue Bild zu spezifizieren.

3.9.2 Beschreibung der Konstruktoren und Destruktoren

3.9.2.1 Client.View.Card.Card (String *s*, int *n*)

Erstellt eine neue Karte fï¿½r die Anzeige und zeichnet dafï¿½r das Bild, das durch die Pfadangabe s und seine KardinaliÃ¤t n im Ordner angegeben ist.

Die Pfadangabe wird durch das Regelwerk bestimmt.

Parameter

s	Pfadangabe zum zu zeichnenden Bild
n	ID der Karte

3.9.3 Dokumentation der Elementfunktionen

3.9.3.1 int Client.View.Card.getID ( )

Gibt die ID der Karte zurÃ¼ck.

Rückgabe

ID der Karte

3.10 Client.View.ChooseCards Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void update (Observable o, Object arg)

3.10.1 Dokumentation der Elementfunktionen

3.10.1.1 void Client.View.ChooseCards.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

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: openChooseCards

3.11 Client.View.ChooseItem Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void update (Observable arg0, Object arg1)

3.11.1 Dokumentation der Elementfunktionen

3.11.1.1 void Client.View.ChooseItem.update (Observable *arg0*, Object *arg1*)

Wird durch notify() im ClientModel aufgerufen.

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

Parameter

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: openChooseItem

3.12 Client.View.CreateGame Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- void addPanelMouseListener (MouseListener m)
- void addLanguageSelectionListener (ItemListener i)
- void addCreateButtonListener (ActionListener a)
- void addLeaveButtonListener (ActionListener a)
- void setLanguage (Language l)
- void update (Observable o, Object arg)

3.12.1 Ausführliche Beschreibung

Es bietet alle Komponenten, um ein Regelwerk zu wï¿½hlen, einen Spielnamen festzulegen und das Spiel durch ein Passwort zu schï¿½tzen. In der Spielerstellung wird ein Titelbild des ausgewï¿½hlten Spiels und eine kurze Beschreibung angezeigt. über 'Leave' kehrt der Spieler in die Lobby zurï¿½ck und mit 'Create' wird das Spiel erstellt.

3.12.2 Dokumentation der Elementfunktionen

3.12.2.1 void Client.View.CreateGame.addPanelMouseListener (MouseListener *m*)

Fügt einen MouseListener zum ImagePanel des CreateGame Fensters hinzu, der zur Anzeige des MouseOver--Texts verwendet wird.

Parameter

m	ein MouseListener

3.12.2.2 void Client.View.CreateGame.addLanguageSelectionListener (ItemListener *i*)

Fügt einen Listener für die Regelwerk-Auswahl des CreateGame Fensters hinzu.

Parameter

i	ein ItemListener

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

Fügt einen ActionListener für den 'Create' Button hinzu.

Parameter

a	ein ActionListener

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

Fügt einen ActionListener für den 'Leave' Button hinzu.

Parameter

a	ein ActionListener

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

Ändert die Sprache des Fensters.

Parameter

l	Sprache in Form des Language-Enums

3.12.2.6 void Client.View.CreateGame.update (Observable *o*, Object *arg*)

Wird durch notify() im ClientModel aufgerufen.

Je nach dem in arg ï¿½bergebenen ViewNotification-Befehl wird ein Update des Fensters ausgefï¿½hrt oder eine Fehlermeldung angezeigt.

Parameter

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: windowChangeAcknowledged, windowChangeDenied

3.13 Client.View.DiscardPile Klassenreferenz

3.14 Client.View.DrawDeck Klassenreferenz

3.15 Client.View.Game Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- Game () throws IOException
- void update (Observable o, Object arg)
- void update (Observable o, String arg)

3.15.1 Ausführliche Beschreibung

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

3.15.2 Beschreibung der Konstruktoren und Destruktoren

3.15.2.1 Client.View.Game.Game ( ) throws IOException

Erstellt das Game Fenster.

Ausnahmebehandlung

IOException	

3.15.3 Dokumentation der Elementfunktionen

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

Wird durch notify() im ClientModel aufgerufen.

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

Parameter

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: chatMessage, playedCardsUpdate, otherDataUpdate

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

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

Parameter

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet eine Chatnachricht in String-Form

3.16 Client.View.GameLobby Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- 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)

3.16.1 Ausführliche Beschreibung

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

3.16.2 Dokumentation der Elementfunktionen

3.16.2.1 void Client.View.GameLobby.addStartButtonListener (ActionListener *a*)

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

Parameter

a	ein ActionListener

3.16.2.2 void Client.View.GameLobby.addRemoveButtonListener (ActionListener *a*)

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

Parameter

a	ein ActionListener

3.16.2.3 void Client.View.GameLobby.addLeaveButtonListener (ActionListener *a*)

Fügt einen ActionListener für den 'Leave' Button hinzu.

Parameter

a	ein ActionListener

3.16.2.4 void Client.View.GameLobby.addChatMessageListener (KeyListener *k*)

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

Parameter

k	

3.16.2.5 void Client.View.GameLobby.setLanguage (Language *l*)

Ändert die Sprache des Fensters.

Parameter

l	Sprache in Form des Language-Enums

3.16.2.6 void Client.View.GameLobby.update (Observable *o*, Object *arg*)

Wird durch notify() im ClientModel aufgerufen.

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

Parameter

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: windowChangeAcknowledged, windowChangeDenied, playerListUpdate, windowChangeForced, chatMessage

3.17 Client.View.GamePanel Klassenreferenz

Abgeleitet von JPanel.

Öffentliche Methoden

- void setupTrickGame (int players)

3.17.1 Ausführliche Beschreibung

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

3.17.2 Dokumentation der Elementfunktionen

3.17.2.1 void Client.View.GamePanel.setupTrickGame (int *players*)

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

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

Parameter

players	Anzahl der Spieler

3.18 Client.View.InputNumber Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void update (Observable o, Object arg)

3.18.1 Dokumentation der Elementfunktionen

3.18.1.1 void Client.View.InputNumber.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

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: openInputNumber

3.19 Client.View.Language Enum-Referenz

3.20 Client.View.Lobby Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- 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)

3.20.1 Ausführliche Beschreibung

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

3.20.2 Dokumentation der Elementfunktionen

3.20.2.1 void Client.View.Lobby.addJoinButtonListener (ActionListener *a*)

Fügt einen ActionListener für den 'Join' Button hinzu.

Parameter

a	ein ActionListener

3.20.2.2 void Client.View.Lobby.addHostButtonListener (ActionListener *a*)

Fügt einen ActionListener für den 'Host' Button hinzu.

Parameter

a	ein ActionListener

3.20.2.3 void Client.View.Lobby.addLeaveButtonListener (ActionListener *a*)

Fügt einen ActionListener für den 'Leave' Button hinzu.

Parameter

a	ein ActionListener

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

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

Parameter

k	

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

Ändert die Sprache des Fensters.

Parameter

l	Sprache in Form des Language-Enums

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

Wird durch notify() im ClientModel aufgerufen.

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

Parameter

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: windowChangeAcknowledged, windowChangeDenied, playerListUpdate, gameListUpdate, chatMessage

3.21 Client.View.Login Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

- void addConnectButtonListener (ActionListener a)
- void addLanguageSelectionListener (ItemListener i)
- void setLanguage (Language l)
- void update (Observable o, Object arg)

3.21.1 Ausführliche Beschreibung

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

3.21.2 Dokumentation der Elementfunktionen

3.21.2.1 void Client.View.Login.addConnectButtonListener (ActionListener *a*)

Fügt einen Listener für den 'Connect' Button des Login Fensters hinzu.

Parameter

a	ein ActionListener

3.21.2.2 void Client.View.Login.addLanguageSelectionListener (ItemListener *i*)

Fügt einen Listener für die Sprachauswahl des Login Fensters hinzu.

Parameter

i	ein ItemListener

3.21.2.3 void Client.View.Login.setLanguage (Language *l*)

Ändert die Sprache des Fensters.

Parameter

l	Sprache in Form des Language-Enums

3.21.2.4 void Client.View.Login.update (Observable *o*, Object *arg*)

Wird durch notify() im ClientModel aufgerufen.

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

Parameter

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: windowChangeAcknowledged, windowChangeDenied

3.22 Client.View.OtherPlayer Klassenreferenz

3.23 Client.View.OwnHand Klassenreferenz

3.23.1 Ausführliche Beschreibung

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

3.24 Client.View.Password Klassenreferenz

Abgeleitet von JFrame und Observer.

Öffentliche Methoden

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

3.24.1 Dokumentation der Elementfunktionen

3.24.1.1 void Client.View.Password.addJoinButtonListener (ActionListener *a*)

Fügt einen ActionListener für den 'Join' Button hinzu.

Parameter

a	ein ActionListener

3.24.1.2 void Client.View.Password.setLanguage (Language *l*)

Ändert die Sprache des Fensters.

Parameter

l	Sprache in Form des Language-Enums

3.24.1.3 void Client.View.Password.update (Observable *o*, Object *arg*)

Wird durch notify() im ClientModel aufgerufen.

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

Parameter

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: windowChangeAcknowledged, windowChangeDenied

3.25 Client.View.ScoreWindow Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void update (Observable o, Object arg)

3.25.1 Dokumentation der Elementfunktionen

3.25.1.1 void Client.View.ScoreWindow.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

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: showScore

3.26 Client.View.ViewCard Klassenreferenz

Abgeleitet von JPanel.

Öffentliche Methoden

- ViewCard (String s, int n)
- int getID ()

3.26.1 Ausführliche Beschreibung

Sie wird verwendet um einzelne Karten auf das Spielfeld zu zeichnen. Dazu enthï¿½lt sie die Pfadangabe zu dem Ordner, in dem die Bilder der Karten gespeichert sind, und eine ID, um das genaue Bild zu spezifizieren.

3.26.2 Beschreibung der Konstruktoren und Destruktoren

3.26.2.1 Client.View.ViewCard.ViewCard (String *s,* int *n*)

Erstellt eine neue Karte fï¿½r die Anzeige und zeichnet dafï¿½r das Bild, das durch die Pfadangabe s und seine KardinaliÃ¤t n im Ordner angegeben ist.

Die Pfadangabe wird durch das Regelwerk bestimmt.

Parameter

s	Pfadangabe zum zu zeichnenden Bild
n	ID der Karte

3.26.3 Dokumentation der Elementfunktionen

3.26.3.1 int Client.View.ViewCard.getID ( )

Gibt die ID der Karte zurÃ¼ck.

Rückgabe

ID der Karte

3.27 Client.View.Warning Klassenreferenz

Abgeleitet von Observer.

Öffentliche Methoden

- void update (Observable o, Object arg)

3.27.1 Ausführliche Beschreibung

Hinweise an, welche vom ClientModel übergeben wurden. Es wird nur im Fehlerfall angezeigt.

3.27.2 Dokumentation der Elementfunktionen

3.27.2.1 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

o	erwartet ein Objekt von der Klasse ClientModel
arg	erwartet: openWarning

3.28 Client.ViewNotification Enum-Referenz

3.29 ComObjects.ComBeenKicked Klassenreferenz

Öffentliche Methoden

- ComBeenKicked (String message)
- String getMessage ()

3.29.1 Ausführliche Beschreibung

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

3.29.2 Beschreibung der Konstruktoren und Destruktoren

3.29.2.1 ComObjects.ComBeenKicked.ComBeenKicked (String *message*)

Dies ist der Kontruktor für eine neue ComBeenKicked-Nachricht.

Parameter

message	ist die Nachricht.

3.29.3 Dokumentation der Elementfunktionen

3.29.3.1 String ComObjects.ComBeenKicked.getMessage ( )

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

Rückgabe

die Nachricht.

3.30 ComObjects.ComChatMessage Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComChatMessage (String message)
- String getChatMessage ()

3.30.1 Ausführliche Beschreibung

Sie enthält eine Chatnachricht in Form eines Strings.

3.30.2 Beschreibung der Konstruktoren und Destruktoren

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

Dies ist der Kontruktor für eine neue ComChatMessage-Nachricht.

Parameter

message	ist die Chatnachricht, die versendet wird.

3.30.3 Dokumentation der Elementfunktionen

3.30.3.1 String ComObjects.ComChatMessage.getChatMessage ( )

Hier kann die versendete Nachricht von anderen Klassen ausgelesen werden.

Rückgabe

die Chatnachricht, die versendet wurde.

3.31 ComObjects.ComClientLeave Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Weitere Geerbte Elemente

3.31.1 Ausführliche Beschreibung

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

3.32 ComObjects.ComClientQuit Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Weitere Geerbte Elemente

3.32.1 Ausführliche Beschreibung

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

3.33 ComObjects.ComCreateGameRequest Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComCreateGameRequest (String name, Enum ruleset, boolean hasPassword, String password)
- String getGameName ()

- Enum getRuleset ()
- boolean hasPassword ()
- String getPassword ()

3.33.1 Ausführliche Beschreibung

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

3.33.2 Beschreibung der Konstruktoren und Destruktoren

3.33.2.1 ComObjects.ComCreateGameRequest.ComCreateGameRequest (String *name,* Enum *ruleset,* boolean *hasPassword,* String *password*)

Dies ist der Kontruktor für eine neue ComCreateGameRequest-Nachricht.

Parameter

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

Benutzt ComObjects.ComCreateGameRequest.hasPassword().

3.33.3 Dokumentation der Elementfunktionen

3.33.3.1 String ComObjects.ComCreateGameRequest.getGameName ( )

Diese Methode gibt den Namen des Spiels zurück.

Rückgabe

den Spielnamen.

3.33.3.2 Enum ComObjects.ComCreateGameRequest.getRuleset ( )

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

Rückgabe

das Regelwerk, welches benutzt wird.

3.33.3.3 boolean ComObjects.ComCreateGameRequest.hasPassword ( )

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

Rückgabe

ob es ein Passwort gibt.

Wird benutzt von ComObjects.ComCreateGameRequest.ComCreateGameRequest().

3.33.3.4 String ComObjects.ComCreateGameRequest.getPassword ( )

Gibt das Passwort zurück.

Sollte keines gesetzt sein, wird null zurück gegeben.

Rückgabe

das Passwort.

3.34 ComObjects.ComInitGameLobby Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComInitGameLobby (List playerList)
- Object getPlayerList ()

3.34.1 Ausführliche Beschreibung

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

3.34.2 Beschreibung der Konstruktoren und Destruktoren

3.34.2.1 ComObjects.ComInitGameLobby.ComInitGameLobby (List *playerList*)

Dies ist der Kontruktor für eine neue ComInitGameLobby-Nachricht.

Parameter

playerList	ist die Liste aller Player, die sich im Wartefenster befinden.

3.34.3 Dokumentation der Elementfunktionen

3.34.3.1 Object ComObjects.ComInitGameLobby.getPlayerList ( )

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

Rückgabe

die Liste der Spieler.

3.35 ComObjects.ComInitLobby Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComInitLobby (List playerList, Set gameList)
- List getPlayerList ()
- Set< GameServerRepresentation > getGameList ()

3.35.1 Ausführliche Beschreibung

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

3.35.2 Beschreibung der Konstruktoren und Destruktoren

3.35.2.1 ComObjects.ComInitLobby.ComInitLobby (List *playerList,* Set *gameList*)

Dies ist der Kontruktor für eine neue ComInitLobby-Nachricht.

Parameter

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

3.35.3 Dokumentation der Elementfunktionen

3.35.3.1 List ComObjects.ComInitLobby.getPlayerList ( )

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

Rückgabe

die Liste der Spieler.

3.35.3.2 Set<GameServerRepresentation> ComObjects.ComInitLobby.getGameList ( )

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

Rückgabe

die Liste der Spiele.

3.36 ComObjects.ComJoinRequest Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComJoinRequest (String gameMasterName)
- String getGameMasterName ()

3.36.1 Ausführliche Beschreibung

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

3.36.2 Beschreibung der Konstruktoren und Destruktoren

3.36.2.1 ComObjects.ComJoinRequest.ComJoinRequest (String *gameMasterName*)

Dies ist der Kontruktor für eine neue ConJoinRequest-Nachricht.

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

Parameter

gameMaster-Name	ist der Name der Spielleiters.

3.36.3 Dokumentation der Elementfunktionen

3.36.3.1 String ComObjects.ComJoinRequest.getGameMasterName ( )

Diese Methode gibt den Namen des Spielleiters zurück.

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

Rückgabe

den Namen des Spielleiters.

3.37 ComObjects.ComKickPlayerRequest Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComKickPlayerRequest (String playerName)
- String getPlayerName ()

3.37.1 Ausführliche Beschreibung

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

3.37.2 Beschreibung der Konstruktoren und Destruktoren

3.37.2.1 ComObjects.ComKickPlayerRequest.ComKickPlayerRequest (String *playerName*)

Dies ist der Kontruktor für eine neue ComKickPlayerRequest-Nachricht.

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

Parameter

playerName	ist der Name des Spielers.

3.37.3 Dokumentation der Elementfunktionen

3.37.3.1 String ComObjects.ComKickPlayerRequest.getPlayerName ( )

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

Rückgabe

den Spielernamen.

3.38 ComObjects.ComLobbyUpdateGamelist Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComLobbyUpdateGamelist (boolean removeFlag, GameServerRepresentation gameServer)
- boolean isRemoveFlag ()
- GameServerRepresentation getGameServer ()

3.38.1 Ausführliche Beschreibung

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

3.38.2 Beschreibung der Konstruktoren und Destruktoren

3.38.2.1 ComObjects.ComLobbyUpdateGamelist.ComLobbyUpdateGamelist (boolean *removeFlag,* GameServerRepresentation *gameServer*)

Dies ist der Kontruktor für eine neue ComLobbyUpdateGamelist-Nachricht.

Parameter

removeFlag	zeigt an, ob das Spiel gelöscht werden soll.
gameServer	ist das Spiel.

3.38.3 Dokumentation der Elementfunktionen

3.38.3.1 boolean ComObjects.ComLobbyUpdateGamelist.isRemoveFlag ( )

Diese Methode liefert, ob ein Spiel gelÃ¶scht werden soll oder nicht.

Rückgabe

ob das Spiel gelöscht wird.

3.38.3.2 GameServerRepresentation ComObjects.ComLobbyUpdateGamelist.getGameServer ( )

Diese Methode liefert das Spiel, das geupdated werden soll.

Rückgabe

das Spiel.

3.39 ComObjects.ComLoginRequest Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComLoginRequest (String name)
- String getPlayerName ()

3.39.1 Ausführliche Beschreibung

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

3.39.2 Beschreibung der Konstruktoren und Destruktoren

3.39.2.1 ComObjects.ComLoginRequest.ComLoginRequest (String *name*)

Dies ist der Kontruktor für eine neue ComLoginRequest-Nachricht.

Parameter

name	ist der Name des Spielers, des sich einloggen mÃ¶chte.

3.39.3 Dokumentation der Elementfunktionen

3.39.3.1 String ComObjects.ComLoginRequest.getPlayerName ( )

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

Dieser muss auf Eindeutigkeit geprüft werden.

Rückgabe

den Spielernamen.

3.40 ComObjects.ComObject Klassenreferenz

Abgeleitet von Serializable.

Basisklasse für 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)

3.40.1 Dokumentation der Elementfunktionen

3.40.1.1 void ComObjects.ComObject.process (**ClientModel** *model*)

Ãœberladene Methode die von dem ClientListenerThread nach empfang einer Nachricht aufgerufen wird.

Parameter

model	Das Client Model, an das sich das ComObjekt weitergibt.

3.40.1.2 void ComObjects.ComObject.process (**Player** *player,* **Server** *server*)

Ãœberladene Methode die von einem PlayerThread nach empfang einer Nachricht aufgerufen wird.

Parameter

player	Der Client welcher den Aufruf startet.
server	Der Server an den sich das ComObjekt weitergibt.

3.41 ComObjects.ComRuleset Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComRuleset (RulesetMessage rulesetMessage)
- RulesetMessage getRulesetMessage ()

3.41.1 Ausführliche Beschreibung

Sie ist die grundlegende Nachricht eines Regelwerkaufrufes und enthält eine verfeinerte Nachricht mit weiteren Informationen, die RulesetMessage.

3.41.2 Beschreibung der Konstruktoren und Destruktoren

3.41.2.1 ComObjects.ComRuleset.ComRuleset (RulesetMessage *rulesetMessage*)

Dies ist der Kontruktor für eine neue ComResult-Nachricht.

Parameter

rulesetMessage	ist eine Nachricht, die ans Ruleset gesendet werden soll.

3.41.3 Dokumentation der Elementfunktionen

3.41.3.1 RulesetMessage ComObjects.ComRuleset.getRulesetMessage ( )

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

Rückgabe

die Nachricht.

3.42 ComObjects.ComServerAcknowledgement Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Weitere Geerbte Elemente

3.43 ComObjects.ComStartGame Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Weitere Geerbte Elemente

3.43.1 Ausführliche Beschreibung

Sie wird versendet, wenn ein Spiel gestartet werden soll.

3.44 ComObjects.ComUpdatePlayerlist Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComUpdatePlayerlist (String playerName, boolean removeFlag)
- String getPlayerName ()
- boolean isRemoveFlag ()

3.44.1 Ausführliche Beschreibung

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

3.44.2 Beschreibung der Konstruktoren und Destruktoren

3.44.2.1 ComObjects.ComUpdatePlayerlist.ComUpdatePlayerlist (String *playerName,* boolean *removeFlag*)

Dies ist der Kontruktor für eine neue ComUpdatePlayerlist-Nachricht.

Diese beinhaltet den Namen des Spielers und die Angabe ob er gelÃ¶scht werden soll.

Parameter

playerName	ist der Name der Spielers.
removeFlag	zeigt, ob der Spieler gelöscht werden soll.

3.44.3 Dokumentation der Elementfunktionen

3.44.3.1 String ComObjects.ComUpdatePlayerlist.getPlayerName ( )

Diese Methode gibt den Namen des Spielers zurück.

Rückgabe

den Spielernamen.

3.44.3.2 boolean ComObjects.ComUpdatePlayerlist.isRemoveFlag ( )

Diese Methode gibt zurück, ob der Spieler aus der Liste gelÃ¶scht werden soll oder nicht.

Rückgabe

ob der Spieler gelöscht werden soll.

3.45 ComObjects.ComWarning Klassenreferenz

Abgeleitet von ComObjects.ComObject.

Öffentliche Methoden

- ComWarning (String warning)
- String getWarning ()

3.45.1 Ausführliche Beschreibung

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

3.45.2 Beschreibung der Konstruktoren und Destruktoren

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

Dies ist der Konstruktor einer neuen ComWarning-Nachricht.

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

Parameter

warning	ist die Warnung, die der Spieler erhält.

3.45.3 Dokumentation der Elementfunktionen

3.45.3.1 String ComObjects.ComWarning.getWarning ( )

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

Rückgabe

die Warnnachricht.

3.46 ComObjects.MsgCard Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Öffentliche Methoden

- MsgCard (Card card)
- Card getCard ()

3.46.1 Ausführliche Beschreibung

Sie beinhaltet die ausgespielte Karte eines Spielers.

3.46.2 Beschreibung der Konstruktoren und Destruktoren

3.46.2.1 ComObjects.MsgCard.MsgCard (**Card** *card*)

Dies ist der Kontruktor für eine neue MsgCard-Nachricht.

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

Parameter

card	ist die Karte.

3.46.3 Dokumentation der Elementfunktionen

3.46.3.1 Card ComObjects.MsgCard.getCard ( )

Diese Methode gibt die ausgespielte Karte des Spielers zurück.

Rückgabe

die Karte.

3.47 ComObjects.MsgCardRequest Klassenreferenz

3.47.1 Ausführliche Beschreibung

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

3.48 ComObjects.MsgGameEnd Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Weitere Geerbte Elemente

3.48.1 Ausführliche Beschreibung

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

3.49 ComObjects.MsgMultiCards Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Öffentliche Methoden

- MsgMultiCards (Set cardList)
- Set< Card > getCardList ()

3.49.1 Ausführliche Beschreibung

Sie liefert mehrere Karten zum Tausch für das Regelwerk Hearts.

3.49.2 Beschreibung der Konstruktoren und Destruktoren

3.49.2.1 ComObjects.MsgMultiCards.MsgMultiCards (Set *cardList*)

Dies ist der Kontruktor für eine neue MsgMultiCards-Nachricht.

Parameter

cardList	ist die Liste der ausgewählten Karten.

3.49.3 Dokumentation der Elementfunktionen

3.49.3.1 Set<Card> ComObjects.MsgMultiCards.getCardList ( )

Gibt die Liste der gewÃ¤hlten Karten zurück.

Rückgabe

die Liste der Karten.

3.50 ComObjects.MsgMultiCardsRequest Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Öffentliche Methoden

- int getCount ()

3.50.1 Dokumentation der Elementfunktionen

3.50.1.1 int ComObjects.MsgMultiCardsRequest.getCount ( )

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

Rückgabe

die Anzahl der Karten.

3.51 ComObjects.MsgMultipleCardsRequest Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Weitere Geerbte Elemente

3.52 ComObjects.MsgNumber Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Öffentliche Methoden

- MsgNumber (int number)
- int getNumber ()

3.52.1 Ausführliche Beschreibung

Sie enthält eine Zahl.

3.52.2 Beschreibung der Konstruktoren und Destruktoren

3.52.2.1 ComObjects.MsgNumber.MsgNumber (int *number*)

Dies ist der Kontruktor für eine neue MsgNumber-Nachricht.

Parameter

number	ist eine Eingabe eines Spielers

3.52.3 Dokumentation der Elementfunktionen

3.52.3.1 int ComObjects.MsgNumber.getNumber ( )

Diese Methode liefert die Eingabe eines Spielers.

Rückgabe

eine Zahl, die Eingabe des Spielers.

3.53 ComObjects.MsgNumberRequest Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Weitere Geerbte Elemente

3.54 ComObjects.MsgSelection Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Öffentliche Methoden

- MsgSelection (int selection)
- int getSelection ()

3.54.1 Ausführliche Beschreibung

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

3.54.2 Beschreibung der Konstruktoren und Destruktoren

3.54.2.1 ComObjects.MsgSelection.MsgSelection (int *selection*)

Dies ist der Kontruktor für eine neue MsgSelection-Nachricht.

Parameter

selection	ist die getroffene Auswahl, reprÃ¤sentiert durch einen Integer.

3.54.3 Dokumentation der Elementfunktionen

3.54.3.1 int ComObjects.MsgSelection.getSelection ( )

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

Rückgabe

die Auswahl.

3.55 ComObjects.MsgSelectionRequest Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Weitere Geerbte Elemente

3.55.1 Ausführliche Beschreibung

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

3.56 ComObjects.MsgUser Klassenreferenz

Abgeleitet von ComObjects.RulesetMessage.

Öffentliche Methoden

- MsgUser (GameClientUpdate gameClientUpdate)
- GameClientUpdate getGameClientUpdate ()

3.56.1 Ausführliche Beschreibung

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

3.56.2 Beschreibung der Konstruktoren und Destruktoren

3.56.2.1 ComObjects.MsgUser.MsgUser (GameClientUpdate *gameClientUpdate*)

Dies ist der Konstruktor einer neuen MsgUser-Nachricht.

Parameter

gameClient-Update	ist der aktuelle Spielstand.

3.56.3 Dokumentation der Elementfunktionen

3.56.3.1 GameClientUpdate ComObjects.MsgUser.getGameClientUpdate ( )

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

Rückgabe

den aktuellen Spielzustand.

3.57 ComObjects.RulesetMessage Klassenreferenz

Abgeleitet von Serializable.

Basisklasse für ComObjects.MsgCard, ComObjects.MsgGameEnd, ComObjects.MsgMultiCards, ComObjects.-MsgMultiCardsRequest, ComObjects.MsgMultipleCardsRequest, ComObjects.MsgNumber, ComObjects.Msg-NumberRequest, ComObjects.MsgSelection, ComObjects.MsgSelectionRequest und ComObjects.MsgUser.

Öffentliche Methoden

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

3.57.1 Ausführliche Beschreibung

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

3.57.2 Dokumentation der Elementfunktionen

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

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

Parameter

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

3.57.2.2 void ComObjects.RulesetMessage.visit (ClientRuleset *clientRuleset*)

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

Parameter

clientRuleset	ist das Ruleset, welches übergeben wird, damit die überladene Methode richtig gewÃ¤hlt wird.

3.58 Ruleset.Card Schnittstellenreferenz

Basisklasse für Ruleset.HeartsCard und Ruleset.WizardCard.

Öffentliche Methoden

- int getValue ()
- Colour getColour ()

3.58.1 Dokumentation der Elementfunktionen

3.58.1.1 int Ruleset.Card.getValue ( )

Gibt den Wert der Karte zurück.

Rückgabe

Der Wert der Karte

Implementiert in Ruleset.WizardCard und Ruleset.HeartsCard.

3.58.1.2 Colour Ruleset.Card.getColour ( )

Gibt die Farbe der Karte zurück.

Rückgabe

Die Farbe der Karte

Implementiert in Ruleset.WizardCard und Ruleset.HeartsCard.

3.59 Ruleset.ClientHearts Klassenreferenz

Abgeleitet von Ruleset.ClientRuleset.

Öffentliche Methoden

- boolean isValidMove (Card card)
- void send (Set< Card > cards)

Geschützte Methoden

- ClientHearts (ClientModel client)

3.59.1 Beschreibung der Konstruktoren und Destruktoren

3.59.1.1 Ruleset.ClientHearts.ClientHearts (ClientModel *client*) [protected]

Erzeugt ein ClientHearts.

Parameter

client	Das Model auf dem gespielt wird

3.59.2 Dokumentation der Elementfunktionen

3.59.2.1 boolean Ruleset.ClientHearts.isValidMove (Card *card*) [virtual]

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

Rückgabe

isValid true falls Zug gültig, false wenn nicht

Implementiert Ruleset.ClientRuleset.

3.59.2.2 void Ruleset.ClientHearts.send (Set< Card > *cards*)

Parameter

cards	

3.60 Ruleset.ClientRuleset Klassenreferenz

Basisklasse für Ruleset.ClientHearts und Ruleset.ClientWizard.

Öffentliche Methoden

- RulesetType getRulesetType ()
- int getMinPlayers ()
- int getMaxPlayers ()
- List< OtherData > getOtherPlayerData ()
- PlayerState getCurrentPlayer ()
- void resolveMessage (RulesetMessage message)
- void send (Card card)

Geschützte Methoden

- ClientRuleset (RulesetType ruleset, int minPlayers, int maxPlayers, ClientModel client)
- void processMessage (MsgUser clientUpdate)
- void processMessage (MsgCardRequest msgCardRequest)
- void processMessage (MsgMultipleCardsRequest msgMultiCardsRequest)
- void processMessage (MsgNumberRequest msgNumber)
- void processMessage (MsgSelectionRequest msgSelection)
- void send (RulesetMessage message)

3.60.1 Ausführliche Beschreibung

Dazu benutzt es die isValidMove() Methode. Des Weiteren kann es vom ClientModel erhaltene RulesetMessages mit der resolveMessage() Methode behandeln.

3.60.2 Beschreibung der Konstruktoren und Destruktoren

3.60.2.1 Ruleset.ClientRuleset.ClientRuleset (**RulesetType** *ruleset,* int *minPlayers,* int *maxPlayers,* **ClientModel** *client*) [protected]

Erstellt eine ClientRuleset Klasse.

Parameter

ruleset	Das Ruleset zum Spiel
minPlayers	Die minimale Spieleranzahl
maxPlayers	Die maximale Spieleranzahl

Benutzt Ruleset.GamePhase.Start.

3.60.3 Dokumentation der Elementfunktionen

3.60.3.1 **RulesetType** Ruleset.ClientRuleset.getRulesetType ( )

Gibt den Typ des Regelwerks zurÃ¼ck.

Rückgabe

Der Typ vom Regelwerk

3.60.3.2 int Ruleset.ClientRuleset.getMinPlayers ( )

Gibt die Mindestanzahl an Spielern zurÃ¼ck fÃ¼r dieses Spiel.

Rückgabe

Die Mindestanzahl an Spielern

3.60.3.3 int Ruleset.ClientRuleset.getMaxPlayers ( )

Gibt die Maximale anzahl an Spielern zurÃ¼ck.

Rückgabe

Die maximale Anzahl an Spielern

3.60.3.4 List<**OtherData**> Ruleset.ClientRuleset.getOtherPlayerData ( )

Holt die Spieldaten der anderen Spieler.

Rückgabe

otherPlayerData Die Spieldaten der anderen Spieler

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

Gibt den Spieler der momentan am Zug ist zurï¿½ck.

Rückgabe

Der momentane Spieler

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

Verarbeitet eine RulesetMessage vom Server.

Parameter

clientUpdate	Die Nachricht vom Server

3.60.3.7 void Ruleset.ClientRuleset.processMessage (MsgUser *clientUpdate*) [protected]

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

Parameter

clientUpdate	Die Nachricht vom Server

3.60.3.8 void Ruleset.ClientRuleset.processMessage (MsgCardRequest *msgCardRequest*) [protected]

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

Parameter

msgCard-Request	Die Nachricht vom Server

3.60.3.9 void Ruleset.ClientRuleset.processMessage (MsgMultipleCardsRequest *msgMultiCardsRequest*) [protected]

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

Parameter

msgMultiCards-Request	Die Nachricht vom Server

3.60.3.10 void Ruleset.ClientRuleset.processMessage (MsgNumberRequest *msgNumber*) [protected]

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

Parameter

msgNumber	Die Nachricht vom Server

3.60.3.11 void Ruleset.ClientRuleset.processMessage (MsgSelectionRequest *msgSelection*) [protected]

Verarbeitet die RulesetMessage dass der Server von dem Spieler verlangt eine Farbe auszuwÃ¤hlen.

Parameter

msgSelection	Die Nachricht vom Server

3.60.3.12 void Ruleset.ClientRuleset.send (Card *card*)

Verpackt eine Karte in ein Rulesetmessage und schick sie an den Server.

Parameter

card	Die Karte

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

Schickt eine RulesetMessage Ã¼bers Model an den Server.

Parameter

message	Die Nachricht

3.61 Ruleset.ClientWizard Klassenreferenz

Abgeleitet von Ruleset.ClientRuleset.

Öffentliche Methoden

- boolean isValidMove (Card card)

Geschützte Methoden

- ClientWizard (ClientModel client)

3.61.1 Beschreibung der Konstruktoren und Destruktoren

3.61.1.1 Ruleset.ClientWizard.ClientWizard (ClientModel *client*) [protected]

Erzeugt ein ClientWizard.

Parameter

client	Das Model auf dem gespielt wird

3.61.2 Dokumentation der Elementfunktionen

3.61.2.1 boolean Ruleset.ClientWizard.isValidMove (Card *card*) [virtual]

Prüft ob ein gemachter Zug zum Spiel Wizard gültig ist.

Rückgabe

isValid true falls Zug gültig, false wenn nicht

Implementiert Ruleset.ClientRuleset.

3.62 Ruleset.Colour Enum-Referenz

3.63 Ruleset.GameClientUpdate Klassenreferenz

Öffentliche Methoden

- List< Card > getOwnHand ()
- Map< String, Card > getPlayedCards ()

- OtherData getOwnData ()
- List< OtherData > getOtherPlayerData ()
- PlayerState getCurrentPlayer ()

3.63.1 Ausführliche Beschreibung

Das wären seine Spielhand, der Ablagestapel sowie die Otherdata von allen Spielern. Bei Wizard enthält es auch die momentane Trumpfkarte.

3.63.2 Dokumentation der Elementfunktionen

3.63.2.1 List<Card> Ruleset.GameClientUpdate.getOwnHand ( )

Holt die Karten die der Client auf der Hand hat.

Rückgabe

ownHand Die Hand des Clients

3.63.2.2 Map<String, Card> Ruleset.GameClientUpdate.getPlayedCards ( )

Holt die gespielten Karten auf dem Ablagestapel.

Rückgabe

discardPile Die gespielten Karten

3.63.2.3 OtherData Ruleset.GameClientUpdate.getOwnData ( )

Holt die zusätzlichen Spieldaten des Client.

Rückgabe

ownData Die Spieldaten des Clients

3.63.2.4 List<OtherData> Ruleset.GameClientUpdate.getOtherPlayerData ( )

Holt die Spieldaten der anderen Spieler.

Rückgabe

otherPlayerData Die Spieldaten der anderen Spieler

3.63.2.5 PlayerState Ruleset.GameClientUpdate.getCurrentPlayer ( )

Gibt den Spieler der momentan am Zug ist zurück.

Rückgabe

Der momentane Spieler

3.64 Ruleset.GamePhase Enum-Referenz

3.65 Ruleset.GameState Klassenreferenz

Öffentliche Methoden

- GameState (RulesetType ruleset, List< Card > deck)

- boolean addPlayerToGame (String name)
- boolean 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 getNumberOfPlayedCards ()
- List< Card > getPlayerCards (String name)
- boolean dealCards (int number)
- boolean giveACard (String name, Card card)
- boolean playCard (Card card)

3.65.1 Ausführliche Beschreibung

Es enthält die einzelnen PlayerStates, sowie Informationen zum Ablage-, Aufnahmestapel, Rundenanzahl, den momentan aktiven Spieler sowie GamePhase.

3.65.2 Beschreibung der Konstruktoren und Destruktoren

3.65.2.1 Ruleset.GameState.GameState (RulesetType *ruleset,* List< Card > *deck*)

Erstellt eine GameStateklasse.

Parameter

ruleset	Der Regelwerktyp des Spiels
deck	Das Kartendeck im Spiel

3.65.3 Dokumentation der Elementfunktionen

3.65.3.1 boolean Ruleset.GameState.addPlayerToGame (String *name*)

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

Parameter

name	

3.65.3.2 boolean Ruleset.GameState.setFirstPlayer (PlayerState *player*)

Setzt einen neuen Spieler als firstPlayer.

Parameter

player	Der neue firstPlayer

3.65.3.3 PlayerState Ruleset.GameState.getFirstPlayer ( )

Holt den Spieler der als erster am Zug war.

Rückgabe

firstPlayer Der Spielzustand des Spielers der als erster am Zug war

3.65.3.4 boolean Ruleset.GameState.setCurrentPlayer (PlayerState *player*)

Setzt einen neuen Spieler als currentPlayer.

Parameter

player	Der neue currentPlayer

3.65.3.5 PlayerState Ruleset.GameState.getCurrentPlayer ( )

Holt den Spieler der momentan am Zug ist.

Rückgabe

currentPlayer Der Spielzustand des Spielers der grad am Zug ist

3.65.3.6 List<Card> Ruleset.GameState.getCardsLeftInDeck ( )

Holt die Karten die noch im Aufnahmestapel sind.

Rückgabe

deck Holt die Karten die noch im Aufnahmestapel sind

3.65.3.7 Map<String,Card> Ruleset.GameState.getPlayedCards ( )

Holt die gespielten Karten im Ablagestapel.

Rückgabe

discardPile Die gespielten Karten

3.65.3.8 PlayerState Ruleset.GameState.getPlayerState (String *name*)

Holt einen bestimmten Spieler.

Parameter

name	Der Name des Spielers

Rückgabe

player Der Spielzustand des Spielers

3.65.3.9 void Ruleset.GameState.setTrumpCard (Card *trumpCard*)

Setzt die Trumpfkarte.

Parameter

trumpCard	Die Trumpfkarte

3.65.3.10 Card Ruleset.GameState.getTrumpCard ( )

Holt die momentane Trumpfkarte im Spiel.

Rückgabe

trumpCard Die momentane Trumpfkarte

3.65.3.11 int Ruleset.GameState.getNumberOfPlayedCards ( )

Holt die Anzahl der gespielten Karten.

Rückgabe

Die Anzahl der gespielten Karten

3.65.3.12 List<Card> Ruleset.GameState.getPlayerCards (String *name*)

Holt die Karten eines Spielers.

Parameter

name	Der Name vom Spieler

Rückgabe

Karten

3.65.3.13 boolean Ruleset.GameState.dealCards (int *number*)

Verteilt eine bestimmte Anzahl an Karten an die Spieler.

Parameter

number	Die Anzahl an Karten

Rückgabe

True falls ein Spieler keine Karten hat

3.65.3.14 boolean Ruleset.GameState.giveACard (String *name,* Card *card*)

Gibt eine bestimmte Karte einem Spieler.

Parameter

name	Der Name des Spielers

Rückgabe

true falls die Karte im Stapel ist, false wenn nicht

3.65.3.15 boolean Ruleset.GameState.playCard (Card *card*)

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

Parameter

card	Die gespielte Karte

Rückgabe

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

3.66 Ruleset.HeartsCard Enum-Referenz

Abgeleitet von Ruleset.Card.

Öffentliche Methoden

- int getValue ()
- Colour getColour ()

3.66.1 Dokumentation der Elementfunktionen

3.66.1.1 int Ruleset.HeartsCard.getValue ( )

Gibt den Wert der Karte zurück.

Rückgabe

Der Wert der Karte

Implementiert Ruleset.Card.

3.66.1.2 Colour Ruleset.HeartsCard.getColour ( )

Gibt die Farbe der Karte zurück.

Rückgabe

Die Farbe der Karte

Implementiert Ruleset.Card.

3.67 Ruleset.HeartsData Klassenreferenz

Abgeleitet von Ruleset.OtherData.

Öffentliche Methoden

- int getCompletePoints ()
- int getCurrentPoints ()

Geschützte Methoden

- void setCompletePoints (int points)
- void setCurrentPoints (int points)

3.67.1 Dokumentation der Elementfunktionen

3.67.1.1 int Ruleset.HeartsData.getCompletePoints ( )

Holt den gesamten Punktestand eines Spielers.

Rückgabe

Der Gesamtpunktstand eines Spielers

3.67.1.2 int Ruleset.HeartsData.getCurrentPoints ( )

Holt den momentanen Punktestand eines Spielers.

Rückgabe

Der momentane Punktestand eines Spielers

3.67.1.3 void Ruleset.HeartsData.setCompletePoints (int *points*) [protected]

/∗∗ Setzt den Gesamt-Punktestand eines Spielers

Parameter

points	Der Gesamt-Punktestand eines Spielers

3.67.1.4 void Ruleset.HeartsData.setCurrentPoints (int *points*) [protected]

/∗∗ Setzt den aktuellen Punktestand eines Spielers

Parameter

points	Der aktuelle Punktestand eines Spielers

3.68 Ruleset.isValidMoveHeartsTest Klassenreferenz

3.69 Ruleset.isValidMoveHeartsTest2_onlyHearts Klassenreferenz

3.70 Ruleset.isValidMoveWizardTest Klassenreferenz

3.71 Ruleset.OtherData Klassenreferenz

Basisklasse für Ruleset.HeartsData und Ruleset.WizData.

Öffentliche Methoden

- OtherData (String name)
- String getname ()

3.71.1 Beschreibung der Konstruktoren und Destruktoren

3.71.1.1 Ruleset.OtherData.OtherData (String *name*)

Erzeugt die zusätzlichen Daten eines Spielers.

Parameter

name	Der Name des Spielers dem die Daten gehören

3.71.2 Dokumentation der Elementfunktionen

3.71.2.1 String Ruleset.OtherData.getname ( )

Holt den Namen des Spielers.

Rückgabe

name Der Name des Spielers

3.72 Ruleset.PlayerState Klassenreferenz

Öffentliche Methoden

- PlayerState (String name, RulesetType ruleset)
- String getName ()
- List< Card > getHand ()
- OtherData getOtherData ()

- void addCard (Card card)
- boolean removeCard (Card card)

3.72.1 Ausführliche Beschreibung

Sie enthält den Namen des Spielers, seine Handkarten und OtherData.

3.72.2 Beschreibung der Konstruktoren und Destruktoren

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

Erstellt einen PlayerState.

Parameter

name	Der Name des Spielers
ruleset	Der Typ des Spiels

3.72.3 Dokumentation der Elementfunktionen

3.72.3.1 String Ruleset.PlayerState.getName ( )

Holt den namen eines Spielers.

Rückgabe

name Der Name des Spielers

3.72.3.2 List<Card> Ruleset.PlayerState.getHand ( )

Holt die Kartenhand des Spielers.

Rückgabe

ownHand Die Kartenhand des Spielers

3.72.3.3 OtherData Ruleset.PlayerState.getOtherData ( )

Holt die zusätzlichen Informationen des Spielers.

Rückgabe

ownHand Die zusätzlichen Informationen des Spielers

3.72.3.4 void Ruleset.PlayerState.addCard (Card *card*)

Gibt dem Spieler eine Karte.

Parameter

card	Die Karte die dem Spieler gegeben wird

3.72.3.5 boolean Ruleset.PlayerState.removeCard (Card *card*)

Entfernt eine Karte aus der Hand des Spielers.

Parameter

card	

Rückgabe

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

3.73 Ruleset.RulesetType Enum-Referenz

3.74 Ruleset.ServerHearts Klassenreferenz

Abgeleitet von Ruleset.ServerRuleset.

Geschützte Methoden

- boolean isValidMove (Card card)

Weitere Geerbte Elemente

3.74.1 Ausführliche Beschreibung

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

3.74.2 Dokumentation der Elementfunktionen

3.74.2.1 boolean Ruleset.ServerHearts.isValidMove (Card *card*) [protected], [virtual]

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

Parameter

card	Die Karte die gespielt wurde
name	Der Name des Spielers

Rückgabe

true falls Zug gültig und false wenn nicht

Implementiert Ruleset.ServerRuleset.

3.75 Ruleset.ServerRuleset Klassenreferenz

Basisklasse für Ruleset.ServerHearts und Ruleset.ServerWizard.

Öffentliche Methoden

- RulesetType getRulesetType ()
- int getMinPlayers ()
- int getMaxPlayers ()
- void resolveMessage (RulesetMessage message, String name)

Geschützte Methoden

- ServerRuleset (RulesetType ruleset, int min, int max, GameServer s)
- boolean setFirstPlayer (PlayerState player)
- PlayerState getFirstPlayer ()
- boolean setCurrentPlayer (PlayerState player)
- PlayerState getCurrentPlayer ()
- void addPlayerToGame (String name)
- PlayerState getPlayerState (String name)
- List< Card > getPlayerCards (String name)
- void send (RulesetMessage message, String name)
- void broadcast (RulesetMessage message)
- void processMessage (MsgCard msgCard, String name)
- void processMessage (MsgMultiCards msgMultiCard, String name)
- void processMessage (MsgNumber msgNumber, String name)
- void processMessage (MsgSelection msgSelection, String name)
- boolean dealCards (int number)
- boolean giveACard (String name, Card card)
- boolean playCard (Card card)
- abstract boolean isValidMove (Card card)

3.75.1 Ausführliche Beschreibung

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

3.75.2 Beschreibung der Konstruktoren und Destruktoren

3.75.2.1 Ruleset.ServerRuleset.ServerRuleset (**RulesetType** *ruleset,* int *min,* int *max,* **GameServer** *s*) [protected]

Erstellt ein ServerRuleset.

Parameter

ruleset	Der Spieltyp

Benutzt Ruleset.GamePhase.Start.

3.75.3 Dokumentation der Elementfunktionen

3.75.3.1 **RulesetType** Ruleset.ServerRuleset.getRulesetType ( )

Gibt den Typ des Regelwerks zurück.

Rückgabe

Der Typ vom Regelwerk

3.75.3.2 int Ruleset.ServerRuleset.getMinPlayers ( )

Gibt die Mindestanzahl an Spielern zurück für dieses Spiel.

Rückgabe

Die Mindestanzahl an Spielern

3.75.3.3 int Ruleset.ServerRuleset.getMaxPlayers ( )

Gibt die Maximale anzahl an Spielern zurück.

Rückgabe

Die maximale Anzahl an Spielern

3.75.3.4 boolean Ruleset.ServerRuleset.setFirstPlayer (PlayerState *player*) [protected]

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

Rückgabe

false wenn der selbe Spieler nochmal als firstPlayer gesetzt wird

3.75.3.5 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

3.75.3.6 boolean Ruleset.ServerRuleset.setCurrentPlayer (PlayerState *player*) [protected]

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

Rückgabe

false wenn der selbe Spieler nochmal als currentPlayer gesetzt wird

3.75.3.7 PlayerState Ruleset.ServerRuleset.getCurrentPlayer ( ) [protected]

Holt den Spieler der gerade am Zug ist.

Rückgabe

currentPlayer Der Spielzustand des Spielers der grad am Zug ist

3.75.3.8 void Ruleset.ServerRuleset.addPlayerToGame (String *name*) [protected]

Fügt einen Spieler ins Spiel ein.

Parameter

name	Der name vom Spieler

3.75.3.9 PlayerState Ruleset.ServerRuleset.getPlayerState (String *name*) [protected]

Holt den Spielerzustand eines Spielers.

Parameter

name	Der Name des Spielers

Rückgabe

playerState Spielzustand eines Spielers

3.75.3.10 List<Card> Ruleset.ServerRuleset.getPlayerCards (String *name*) [protected]

Holt die Spielkarten eines Spielers.

Parameter

name	Der Name eines Spielers

Rückgabe

Die Spielkarten des Spielers

3.75.3.11 void Ruleset.ServerRuleset.send (RulesetMessage *message*, String *name*) [protected]

Schickt eine Nachricht an einen Spieler.

Parameter

message	Die Nachricht vom Typ RulesetMessage
name	Der Name vom Spieler

3.75.3.12 void Ruleset.ServerRuleset.broadcast (RulesetMessage *message*) [protected]

Schickt eine Nachricht an alle Spieler.

Parameter

message	Die Nachricht

3.75.3.13 void Ruleset.ServerRuleset.resolveMessage (RulesetMessage *message*, String *name*)

Verarbeitet eine RulesetMessage die von einem Spieler kommt.

Parameter

message	Die Nachricht
name	Der Name des Spielers

3.75.3.14 void Ruleset.ServerRuleset.processMessage (MsgCard *msgCard*, String *name*) [protected]

Verarbeitet die RulesetMessage dass eine Karte vom Spieler gespielt.

Parameter

msgCard	Die Nachricht vom Client welche Karte gespielt wurde
name	Der Name des Spielers

3.75.3.15 void Ruleset.ServerRuleset.processMessage (MsgMultiCards *msgMultiCard*, String *name*) [protected]

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

Parameter

msgMultiCard	Die Nachricht vom Client
name	Der Name des Spielers

3.75.3.16 void Ruleset.ServerRuleset.processMessage (MsgNumber *msgNumber,* String *name*) [protected]

Verarbeitet die RulesetMessage dass ein Spieler eine Stichangabe gemacht hat.

Parameter

msgNumber	Die Nachricht vom Client
name	Der Name des Spielers

3.75.3.17 void Ruleset.ServerRuleset.processMessage (MsgSelection *msgSelection,* String *name*) [protected]

Verarbeitet die RulesetMessage dass ein Spieler eine Farbe ausgewählt hat.

Parameter

msgSelection	Die Nachricht vom Client
name	Der Name des Spielers

3.75.3.18 boolean Ruleset.ServerRuleset.dealCards (int *number*) [protected]

Verteil eine bestimmte Anzahl an Karten an die Spieler.

Parameter

number	Die Anzahl an Karten

Rückgabe

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

3.75.3.19 boolean Ruleset.ServerRuleset.giveACard (String *name,* Card *card*) [protected]

Gibt einem Spieler eine bestimmte Karte.

Parameter

name	Der Name eines Spielers

Rückgabe

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

3.75.3.20 boolean Ruleset.ServerRuleset.playCard (Card *card*) [protected]

Der momentane Spieler spielt eine Karte.

Parameter

card	Die gespielte Karte

Rückgabe

true falls der Spieler die Karte hat

3.75.3.21 abstract boolean Ruleset.ServerRuleset.isValidMove (Card *card*) [protected], [pure virtual]

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

Parameter

card	Die Karte die gespielt wurde
name	Der Name des Spielers

Rückgabe

true falls Zug gültig und false wenn nicht

Implementiert in Ruleset.ServerWizard und Ruleset.ServerHearts.

3.76 Ruleset.ServerWizard Klassenreferenz

Abgeleitet von Ruleset.ServerRuleset.

Geschützte Methoden

- boolean isValidMove (Card card)

Weitere Geerbte Elemente

3.76.1 Ausführliche Beschreibung

Sie enthält zudem weitere Methoden, welche für das Spiel Wizard spezifisch benötigt werden, wie das Bestimmen einer Trumpffarbe und die Bestimmung der Rundenanzahl.

3.76.2 Dokumentation der Elementfunktionen

3.76.2.1 boolean Ruleset.ServerWizard.isValidMove (Card *card*) [protected], [virtual]

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

Parameter

card	Die Karte die gespielt wurde
name	Der Name des Spielers

Rückgabe

true falls Zug gültig und false wenn nicht

Implementiert Ruleset.ServerRuleset.

3.77 Ruleset.TestHeartsWinner Klassenreferenz

3.78 Ruleset.TestWizardWinner Klassenreferenz

3.79 Ruleset.WizardCard Enum-Referenz

Abgeleitet von Ruleset.Card.

Öffentliche Methoden

- int getValue ()
- Colour getColour ()

3.79.1 Dokumentation der Elementfunktionen

3.79.1.1 int Ruleset.WizardCard.getValue ( )

Gibt den Wert der Karte zurück.

Rückgabe

Der Wert der Karte

Implementiert Ruleset.Card.

3.79.1.2 Colour Ruleset.WizardCard.getColour ( )

Gibt die Farbe der Karte zurück.

Rückgabe

Die Farbe der Karte

Implementiert Ruleset.Card.

3.80 Ruleset.WizData Klassenreferenz

Abgeleitet von Ruleset.OtherData.

Öffentliche Methoden

- int getAnnouncedTricks ()
- int getAchievedTricks ()
- int getPoints ()
- void announceTricks (int annouceTricks)

Geschützte Methoden

- void setPoints (int points)

3.80.1 Dokumentation der Elementfunktionen

3.80.1.1 int Ruleset.WizData.getAnnouncedTricks ( )

Holt die angesagten Stiche des Spielers.

Rückgabe

announcedTricks Die angesagten Stiche

3.80.1.2 int Ruleset.WizData.getAchievedTricks ( )

Holt die erreichten Stiche des Spielers.

Rückgabe

achievedTricks Die gemachten Stiche eines Spielers

3.80.1.3 int Ruleset.WizData.getPoints ( )

Holt den Punktestand des Spielers.

Rückgabe

points Der Punktestand des Spielers

3.80.1.4 void Ruleset.WizData.announceTricks (int *annouceTricks*)

Beim Spielstart werden die vorausgesagten Stiche des Spieler gespeichert und die gemachten Stiche zurückgesetzt.

Parameter

annouceTricks	Die vorausgesagten Stiche des Spielers

3.80.1.5 void Ruleset.WizData.setPoints (int *points*) [protected]

Setzt den Punktestand eines Spielers.

Parameter

points	Der Punktestand eines Spielers

3.81 Server.ClientListenerThread Klassenreferenz

Abgeleitet von Runnable.

3.82 Server.GameServer Klassenreferenz

Abgeleitet von Server.Server.

Öffentliche Methoden

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

3.82.1 Ausführliche Beschreibung

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

Autor

Viktoria

3.82.2 Beschreibung der Konstruktoren und Destruktoren

3.82.2.1 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 übergebenen Werte. Setzt den gameMasterName auf den Namen des gameMaster und fügt den gameMaster dem Set an Spielern hinzu. Bestimmt mithilfe des Enums RulesetType das Ruleset und erstellt es. Setzt currentPlayers auf eins und maxPlayers je nach Ruleset.

Parameter

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

3.82.3 Dokumentation der Elementfunktionen

3.82.3.1 synchronized void Server.GameServer.addPlayer (Player *player*)

Diese Methode wird vom abstrakten Server vererbt.

Zusätzlich wird die Zahl der currentPlayers um eins Erhöht.

Parameter

player	ist der Player, der hinzugefoügt wird

3.82.3.2 synchronized void Server.GameServer.removePlayer (Player *player*)

Diese Methode wird vom abstrakten Server vererbt.

Zusätzlich wird die Zahl der currentPlayers um eins Verringert.

Parameter

player	ist der Player, der entfernt wird

3.82.3.3 void Server.GameServer.receiveMessage (Player *player,* ComKickPlayerRequest *kickPlayer*) throws IOException

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

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

Parameter

player	ist der Thread der die Nachricht erhalten hat
kicked	ist das ComObject, das verarbeitet wird

Ausnahmebehandlung

IOException	

3.82.3.4 void Server.GameServer.receiveMessage (Player *player,* ComChatMessage *chat*) throws IOException

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

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

Parameter

player	ist der Thread der die Nachricht erhalten hat
chat	ist das ComObject, das die Chatnachricht enthält

Ausnahmebehandlung

IOException	

Benutzt Server.Server.broadcast().

3.82.3.5 void Server.GameServer.receiveMessage (Player *player,* ComClientLeave *leave*) throws IOException

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

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

Parameter

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

Ausnahmebehandlung

IOException	

3.82.3.6 void Server.GameServer.receiveMessage (Player *player,* ComClientQuit *quit*) throws IOException

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

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

Parameter

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

Ausnahmebehandlung

IOException	

3.82.3.7 void Server.GameServer.receiveMessage (Player *player,* ComStartGame *start*)

Diese Methode sagt dem Ruleset, dass ein neues Spiel gestartet werden soll.

Parameter

player	ist der Thread der die Nachricht erhalten hat

start	ist das ComObject, dass angibt, dass das Spiel gestartet werden soll

3.82.3.8 void Server.GameServer.receiveMessage (Player *player,* ComRuleset *ruleset*)

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

Parameter

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

3.82.3.9 ComInitGameLobby Server.GameServer.initLobby ( )

Baut ein neues ComInitGameLobby Objekt und gibt es zurück.

Rückgabe

Gibt das ComInitGameLobby Objekt zurück

3.83 Server.GameServerRepresentation Klassenreferenz

Öffentliche Methoden

- GameServerRepresentation (String gameMaster, String gameName, int max, int current, RulesetType type, boolean password)

3.83.1 Ausführliche Beschreibung

Sie wird dem ComObjekt ComLobbyUpdateGameList angehängt, um die Spielliste in der GameLobby aktualisieren zu können

Autor

Viktoria

3.83.2 Beschreibung der Konstruktoren und Destruktoren

3.83.2.1 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

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

3.84 Server.LobbyServer Klassenreferenz

Abgeleitet von Server.Server.

Klassen

- class ClientListenerThread

Öffentliche Methoden

- void receiveMessage (Player player, ComChatMessage chat) throws IOException
- void receiveMessage (Player player, ComClientQuit quit) throws IOException
- void receiveMessage (Player player, ComCreateGameRequest create) throws IOException
- void receiveMessage (Player player, ComJoinRequest join) throws IOException
- void receiveMessage (Player player, ComLoginRequest login) throws IOException
- ComInitLobby initLobby ()
- Set< GameServer > getGameServerSet ()

3.84.1 Ausführliche Beschreibung

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

Autor

Viktoria

3.84.2 Dokumentation der Elementfunktionen

3.84.2.1 void Server.LobbyServer.receiveMessage (Player *player,* ComChatMessage *chat*) throws IOException

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

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

Parameter

player	ist der Thread der die Nachricht erhalten hat
chat	ist das ComObject, das die Chatnachricht enthält

Ausnahmebehandlung

IOException	

Benutzt Server.Server.broadcast().

3.84.2.2 void Server.LobbyServer.receiveMessage (Player *player,* ComClientQuit *quit*) throws IOException

Diese 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

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

Ausnahmebehandlung

IOException	

3.84.2.3 void Server.LobbyServer.receiveMessage (Player *player,* ComCreateGameRequest *create*) throws IOException

Diese Methode erstellt einen neuen GameServer fügt ihm den Player hinzu.

Durch broadcast wird sowohl im LobbyServer als auch im GameServer ein ComUpdatePlayerlist verschickt. Zusätzlich wird dem Client mit sendToPlayer ein ComInitGameLobby geschickt.

Parameter

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

Ausnahmebehandlung

IOException	

3.84.2.4 void Server.LobbyServer.receiveMessage (Player *player,* ComJoinRequest *join*) throws IOException

Diese Methode fügt einen Player dem entsprechenden GameServer hinzu.

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

Parameter

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

Ausnahmebehandlung

IOException	

3.84.2.5 void Server.LobbyServer.receiveMessage (Player *player,* ComLoginRequest *login*) throws IOException

Diese Methode überprüft, ob der Name im Set names vorhanden ist, falls ja, wird ein ComWarning an den Client geschickt, falls nein, wird im Player setName aufgerufen.

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

Parameter

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

Ausnahmebehandlung

IOException	

3.84.2.6 ComInitLobby Server.LobbyServer.initLobby ( )

Baut ein neues ComInitLobby Objekt und gibt es zurück.

Rückgabe

Gibt das ComInitLobby Objekt zurück

3.84.2.7 Set<GameServer> Server.LobbyServer.getGameServerSet ( )

Getter für das GameServerSet.

Rückgabe

Gibt das gameServerSet zurück

3.85 Server.LobbyServer.ClientListenerThread Klassenreferenz

Abgeleitet von Runnable.

3.85.1 Ausführliche Beschreibung

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

Autor

Viktoria

3.86 Server.LobbyServerTest Klassenreferenz

3.87 Server.Player Klassenreferenz

Abgeleitet von Runnable.

Öffentliche Methoden

- Player (Server lobbyServer, ObjectOutputStream output, ObjectInputStream input)
- void send (ComObject com) throws IOException
- void changeServer (Server newServer) throws IOException
- String getName ()
- void setName (String newName)

3.87.1 Ausführliche Beschreibung

Sie verwaltet für die Dauer einer Serververbindung die Verbindung zum Client

Autor

Viktoria

3.87.2 Beschreibung der Konstruktoren und Destruktoren

3.87.2.1 Server.Player.Player (Server *lobbyServer,* ObjectOutputStream *output,* ObjectInputStream *input*)

Konstruktor des Players, in ihm werden die Attribute server, comOut und ComIn mit vom ClientListererThread übergebenen werten Instanziiert.

Parameter

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

3.87.3 Dokumentation der Elementfunktionen

3.87.3.1 void Server.Player.send (ComObject *com*) throws IOException

Diese Methode schickt ein ComObjekt an den Client.

Parameter

com	ist das ComObject das verschickt wird

Ausnahmebehandlung

IOException	wenn der Output nicht funktioniert

3.87.3.2 void Server.Player.changeServer (Server *newServer*) throws IOException

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 eingefugt. Danach wird vom neuen Server ein ComUpdatePlayerlist Objekt mit broadcast an alle Clients, die vom Server verwaltet werden, verschickt.

Parameter

newServer	ist der neue Server

Ausnahmebehandlung

IOException	wenn der Output nicht funktioniert

Benutzt Server.Player.getName().

3.87.3.3 String Server.Player.getName ( )

Getter-Methode für den Benutzernamen.

Rückgabe

gibt den Benutzernamen des Spielers zurück

Wird benutzt von Server.Player.changeServer().

3.87.3.4 void Server.Player.setName (String *newName*)

Setter-Methode für den Benutzernamen.

Parameter

newName	ist der neue Name

3.88 Server.QuitGameTest Klassenreferenz

Abgeleitet von TestCase.

3.89 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) throws IOException
- synchronized void addPlayer (Player player)
- synchronized void removePlayer (Player player)
- synchronized void broadcast (ComObject com) throws IOException

3.89.1 Ausführliche Beschreibung

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

Autor

Viktoria

3.89.2 Dokumentation der Elementfunktionen

3.89.2.1 void Server.Server.receiveMessage (Player *player,* ComObject *com*)

Diese Methode dient zur Verarbeitung von eingehenden ComObjects.

Parameter

player	ist der Player von dem die Nachricht kommt
com	ist das ComObjekt vom Client verschickt wurde

3.89.2.2 synchronized void Server.Server.sendToPlayer (String *name,* ComObject *com*) throws IOException

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

Der Player der die Nachricht verschicken soll wird Anhand des übergebenen Benutzernamens identifiziert. Ist der Name oder das ComObject leer wird nichts verschickt.

Parameter

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

Ausnahmebehandlung

IOException	wenn der Output nicht funktioniert

Benutzt Server.Server.playerSet.

3.89.2.3 synchronized void Server.Server.addPlayer (Player *player*)

Diese Methode fügt einen Player dem Set an Playern hinzu, welche der Server verwaltet.

Parameter

player	ist der Player, der hinzugefoügt wird

3.89.2.4 synchronized void Server.Server.removePlayer (Player *player*)

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

Parameter

player	ist der Player, der entfernt wird

3.89.2.5 synchronized void Server.Server.broadcast (ComObject *com*) throws IOException

Diese Methode wird genutzt, um ein ComObject an alle Clients, die vom Server verwaltet werden, zu schicken.

Ist das ComObject leer, passiert nichts.

Parameter

com	ist das ComObject, dass verschickt werden soll

Ausnahmebehandlung

IOException	wenn der Output nicht funktioniert

Benutzt Server.Server.playerSet.

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

3.90 Server.ServerMain Klassenreferenz

Öffentliche, statische Methoden

- static void main (String[] args)

3.90.1 Ausführliche Beschreibung

Autor

Viktoria

3.90.2 Dokumentation der Elementfunktionen

3.90.2.1 static void Server.ServerMain.main (String[] *args*) [static]

Die main-Methode erstellt einen neuen LobbyServer.

Parameter

args	

4 JUnit-Tests

JUnit-Test werden für die KlassenClientModel, LobbyServer, GameServer, sowie den ClientRulesets und Server-Rulesets erstellt. Folgende Klassen Testfälle sind bereits implementiert:

4.1 isValidWizardMove

```
package Ruleset;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;
```

```
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;

public class isValidMoveWizardTest {
   String player1 = "Hans";
   String player2 = "Josef";
   String player3 = "Joe";

   @Mock
   ServerRuleset ruleset;

      @Before
      public void setUpGame() {
            ruleset.initGame();
            ruleset.addPlayerToGame(player1);
            ruleset.addPlayerToGame(player2);
            ruleset.addPlayerToGame(player3);

            ruleset.setFirstPlayer(ruleset.getPlayerState(player1));
            ruleset.setTrumpCard(WizardCard.VierRot);

            ruleset.giveACard(player1, WizardCard.DreiGruen);
            ruleset.giveACard(player1, WizardCard.ZaubererRot);
            ruleset.givaACard(player1, WizardCard.ZweiBlau);

            ruleset.giveACard(player2, WizardCard.ZweiGruen);
            ruleset.giveACard(player2, WizardCard.DreiRot);
            ruleset.givaACard(player2, WizardCard.ZweiGelb);

            ruleset.giveACard(player3, WizardCard.NarrBlau);
            ruleset.giveACard(player3, WizardCard.EinsGruen);
            ruleset.giveACard(player3, WIzardCard.ZweiRot);

      }

      public void testSorcerer() {
            ruleset.playCard(WizardCard.ZaubererRot);
            ruleset.setCurrentPlayer(ruleset.getPlayerState(player2));

            boolean isValidMove = ruleset.isValidMove(WizardCard.DreiRot);

            assertTrue(isValidMove);
      }

      @Test
      public void testRed3OnGreen3() {
            ruleset.playCard(WizardCard.DreiGruen);
            ruleset.setCurrentPlayer(ruleset.getPlayerState(player2));
            boolean isValidMove = ruleset.isValidMove(WizardCard.DreiRot);

            assertFalse(isValidMove);
      }

      @Test
      public void testGreen2OnGreen3() {
            ruleset.playCard(WizardCard.DreiGruen);
            ruleset.setCurrentPlayer(ruleset.getPlayerState(player2));

            boolean isValidMove = ruleset.isValidMove(WizardCard.ZweiGruen);

            assertTrue(isValidMove);
      }
```

```
	@Test
	public void testFoolBlueOnGreen2OnGreen3() {
		ruleset.playCard(WizardCard.DreiGruen);
		ruleset.setCurrentPlayer(ruleset.getPlayerState(player2));

		ruleset.playCard(WizardCard.ZweiGruen);
		ruleset.setCurrentPlayer(ruleset.getPlayerState(player3));

		boolean isValidMove = ruleset.isValidMove(WizardCard.NarrBlau);

		assertTrue(isValidMove);
	}

}
```

4.2 isValidHeartsMove

```
package Ruleset;

import static org.junit.Assert.*;

import java.util.ArrayList;
import java.util.List;

import org.junit.Test;
import org.junit.runners.JUnit4;

public class isValidMoveHeartsTest {
   @Test
   public void testIsValidMove() {
      GameServer
      ServerRuleset server = new ServerHearts();
      String player1 = "Hans";
      String player2 = "Josef";
      String player3 = "Joe";
      String player4 = "Horst";
      server.initGame();
      server.addPlayerToGame(player1);
      server.addPlayerToGame(player2);
      server.addPlayerToGame(player3);
      server.addPlayerToGame(player4);

      server.setFirstPlayer(server.getPlayerState(player1));

      server.giveACard(player1, HeartsCard.Herz2);
      server.giveACard(player1, HeartsCard.Kreuz9);
      server.giveACard(player2, HeartsCard.Caro3);
      server.giveACard(player2, HeartsCard.Caro6);
      server.giveACard(player3, HeartsCard.Pik4);
      server.giveACard(player3, HeartsCard.Pik5);
      server.giveACard(player4, HeartsCard.Pik1);
      server.giveACard(player4, HeartsCard.Herz7);

      boolean isValidMove = server.isValidMove(HeartsCard.Herz2);

      assertEquals(isValidMove, false);

      boolean isValidMove2 = server.isValidMove(HeartsCard.Caro3);

      assertEquals(isValidMove2, true);
```

```
    }
}
```

```
package Ruleset;

import static org.junit.Assert.*;

import java.util.ArrayList;
import java.util.List;

import org.junit.Test;
import org.junit.runners.JUnit4;

public class isValidMoveHeartsTest2_onlyHearts {

    @Test
    public void testIsValidMove() {
        ServerRuleset server = new ServerHearts();
        String player1 = "Hans";
        String player2 = "Josef";
        String player3 = "Joe";
        String player4 = "Horst";
        server.initGame();
        server.addPlayerToGame(player1);
        server.addPlayerToGame(player2);
        server.addPlayerToGame(player3);
        server.addPlayerToGame(player4);

        server.setFirstPlayer(server.getPlayerState(player1));

        server.giveACard(player1, HeartsCard.Herz2);
        server.giveACard(player1, HeartsCard.Herz5);
        server.giveACard(player2, HeartsCard.Caro3);
        server.giveACard(player2, HeartsCard.Caro6);
        server.giveACard(player3, HeartsCard.Pik4);
        server.giveACard(player3, HeartsCard.Pik5);
        server.giveACard(player4, HeartsCard.Pik1);
        server.giveACard(player4, HeartsCard.Herz7);

        boolean isValidMove = server.isValidMove(HeartsCard.Herz2);

        assertEquals(isValidMove, true);
    }
}
```

4.3 TestWinner

```
package Ruleset;

import static org.junit.Assert.*;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

import org.junit.Test;

import Server.GameServer;
import Server.LobbyServer;
```

```
import Server.Player;

public class TestHeartsWinner {

	@Test
	public void testGetWinner() throws IOException {
		LobbyServer lserver = new LobbyServer();
		Socket socket1 = new Socket();
		Socket socket2 = new Socket();
		Socket socket3 = new Socket();
		Socket socket4 = new Socket();
		ObjectOutputStream out1 = new
		    ObjectOutputStream(socket1.getOutputStream());
		out1.flush();
		ObjectOutputStream out2 = new
		    ObjectOutputStream(socket2.getOutputStream());
		out2.flush();
		ObjectOutputStream out3 = new
		    ObjectOutputStream(socket3.getOutputStream());
		out3.flush();
		ObjectOutputStream out4 = new
		    ObjectOutputStream(socket4.getOutputStream());
		out4.flush();
		ObjectInputStream in1 = new ObjectInputStream(socket1.getInputStream());
		ObjectInputStream in2 = new ObjectInputStream(socket2.getInputStream());
		ObjectInputStream in3 = new ObjectInputStream(socket3.getInputStream());
		ObjectInputStream in4 = new ObjectInputStream(socket4.getInputStream());
		Player blue = new Player(lserver, out1, in1);
		Player white = new Player(lserver, out2, in2);
		Player orange = new Player(lserver, out3, in3);
		Player brown = new Player(lserver, out4, in4);
		GameServer server = new GameServer(lserver, blue, "Some Game",
		    RulesetType.Hearts, "", false);
		server.addPlayer(white);
		server.addPlayer(orange);
		server.addPlayer(brown);
		ServerHearts hearts = new ServerHearts(server);
		hearts.addPlayerToGame("Mr. Blue");
		hearts.addPlayerToGame("Mr. White");
		hearts.addPlayerToGame("Mr. Orange");
		hearts.addPlayerToGame("Mr. Brown");

		OtherData dateblue = hearts.getPlayerState("Mr. Blue").getOtherData();
		HeartsData heartsdatblue = (HeartsData) dateblue;
		heartsdatblue.setCompletePoints(80);

		OtherData datewhite = hearts.getPlayerState("Mr. White").getOtherData();
		HeartsData heartsdatwhite = (HeartsData) datewhite;
		heartsdatwhite.setCompletePoints(200);

		OtherData dateorange = hearts.getPlayerState("Mr.
		    Orange").getOtherData();
		HeartsData heartsdatorange = (HeartsData) dateorange;
		heartsdatorange.setCompletePoints(130);

		OtherData datebrown = hearts.getPlayerState("Mr. Brown").getOtherData();
		HeartsData heartsdatbrown = (HeartsData) datebrown;
		heartsdatbrown.setCompletePoints(240);

		assertTrue(hearts.getWinner().compareTo("Mr. Brown") == 0);
	}

}
```

```
package Ruleset;

import static org.junit.Assert.*;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

import org.junit.Test;

import ComObjects.ComChatMessage;
import ComObjects.ComObject;
import Server.GameServer;
import Server.LobbyServer;
import Server.Player;

/**
 * Testet ob der richtige Sieger ermittelt wird und ob jedem Mitspieler
 * der richtige Sieger mitgeteilt wird
 */
public class TestWizardWinner {

	@Test
	public void testGetWinner() throws IOException{
		LobbyServer lserver = new LobbyServer();
		Socket socket1 = new Socket();
		Socket socket2 = new Socket();
		Socket socket3 = new Socket();
		Socket socket4 = new Socket();
		ObjectOutputStream out1 = new
		    ObjectOutputStream(socket1.getOutputStream());
		out1.flush();
		ObjectOutputStream out2 = new
		    ObjectOutputStream(socket2.getOutputStream());
		out2.flush();
		ObjectOutputStream out3 = new
		    ObjectOutputStream(socket3.getOutputStream());
		out3.flush();
		ObjectOutputStream out4 = new
		    ObjectOutputStream(socket4.getOutputStream());
		out4.flush();
		ObjectInputStream in1 = new ObjectInputStream(socket1.getInputStream());
		ObjectInputStream in2 = new ObjectInputStream(socket2.getInputStream());
		ObjectInputStream in3 = new ObjectInputStream(socket3.getInputStream());
		ObjectInputStream in4 = new ObjectInputStream(socket4.getInputStream());
		Player blue = new Player(lserver, out1, in1);
		Player white = new Player(lserver, out2, in2);
		Player orange = new Player(lserver, out3, in3);
		Player brown = new Player(lserver, out4, in4);
		GameServer server = new GameServer(lserver, blue, "Some Game",
		    RulesetType.Wizard, "", false);
		server.addPlayer(white);
		server.addPlayer(orange);
		server.addPlayer(brown);
		ServerWizard wiz = new ServerWizard(server);
		wiz.addPlayerToGame("Mr. Blue");
		wiz.addPlayerToGame("Mr. White");
```

```
            wiz.addPlayerToGame("Mr. Orange");
            wiz.addPlayerToGame("Mr. Brown");

            OtherData dateblue = wiz.getPlayerState("Mr. Blue").getOtherData();
            WizData wizdatblue = (WizData) dateblue;
            wizdatblue.setPoints(80);

            OtherData datewhite = wiz.getPlayerState("Mr. White").getOtherData();
            WizData wizdatwhite = (WizData) datewhite;
            wizdatwhite.setPoints(200);

            OtherData dateorange = wiz.getPlayerState("Mr. Orange").getOtherData();
            WizData wizdatorange = (WizData) dateorange;
            wizdatorange.setPoints(130);

            OtherData datebrown = wiz.getPlayerState("Mr. Brown").getOtherData();
            WizData wizdatbrown = (WizData) datebrown;
            wizdatbrown.setPoints(240);

            assertTrue(wiz.getWinner().compareTo("Mr. Brown") == 0);
      }

}
```

4.4 Chattest

```
package Client;

import static org.junit.Assert.*;

import org.junit.Test;
import org.junit.After;
import org.junit.Before;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;
import ComObjects.ComChatMessage;

public class ClientModelChatTest {

      ComChatMessage testMessage;

      ClientModel testModel;

      @Mock
      MessageListenerThread netIO;

      @Before
   public void setUp() {
            MockitoAnnotations.initMocks(this);
            testMessage = new ComChatMessage("Hello Test!");
            testModel = new ClientModel();
   }

   @After
   public void tearDown() {
      netIO = null;
      testMessage = null;
      testModel = null;
   }
```

```
	@Test
	public void testSendChatMessage() {
		testModel.setNetIO(netIO);
		testModel.sendMessage(testMessage);
		Mockito.verify(netIO).send(testMessage);
	}

	@Test
	public void testReceiveChatMessage() {
		testMessage.process(testModel);
		assertTrue(testModel.getChatMessage().equals(testMessage.getChatMessage()));
	}

}
```

```
package Server;

import java.io.IOException;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;

import ComObjects.ComChatMessage;

public class LobbyServerTest {

	ComChatMessage testMessage;

	LobbyServer testServer;

	@Mock
	Player player1;

	@Mock
	Player player2;

	@Mock
	Player player3;

	@Before
	public void setUp() {
		MockitoAnnotations.initMocks(this);
		testMessage = new ComChatMessage("Hello Test!");
		testServer = new LobbyServer();
	}

	@After
	public void tearDown() {
		testMessage = null;
		testServer = null;
		player1 = null;
		player2 = null;
		player3 = null;
	}

	@Test
	public void testReceiveMessagePlayerComChatMessage() throws IOException {
		testServer.addPlayer(player1);
		testServer.addPlayer(player2);
```

```
            testServer.addPlayer(player3);
            testMessage.process(player1, testServer);
            Mockito.verify(player1).send(testMessage);
            Mockito.verify(player2).send(testMessage);
            Mockito.verify(player3).send(testMessage);
      }
}
```

4.5 QuitGameTest

```
package Server;

import java.io.BufferedInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;


import org.junit.Test;

import ComObjects.ComClientQuit;
import Ruleset.RulesetType;

import junit.framework.*;

public class QuitGameTest extends TestCase{

      public QuitGameTest(String name){
            super(name);
      }
      @Test
      public void testPlayerQuitGame() throws IOException{
            LobbyServer lobby = new LobbyServer();
            Player player1 = new Player(lobby, new ObjectOutputStream(System.out),
                new ObjectInputStream(new BufferedInputStream(System.in)));
            player1.setName("MrBlue");
            lobby.addPlayer(player1);
            GameServer game = new GameServer(lobby, player1, "MrBluesGame",
                RulesetType.Hearts, null, false);

            player1.changeServer(game);
            assertTrue(game.playerSet.contains(player1));

            Player player2 = new Player(lobby, new ObjectOutputStream(System.out),
                new ObjectInputStream(new BufferedInputStream(System.in)));
            player2.setName("MrWhite");
            Player player3 = new Player(lobby, new ObjectOutputStream(System.out),
                new ObjectInputStream(new BufferedInputStream(System.in)));
            player3.setName("MrPink");
            Player player4 = new Player(lobby, new ObjectOutputStream(System.out),
                new ObjectInputStream(new BufferedInputStream(System.in)));
            player4.setName("MrRed");

            game.addPlayer(player2);
            game.addPlayer(player3);
            game.addPlayer(player4);

            ComClientQuit quit = new ComClientQuit();
            game.receiveMessage(player1, quit);

            assertFalse(lobby.getGameServerSet().contains(game));
```

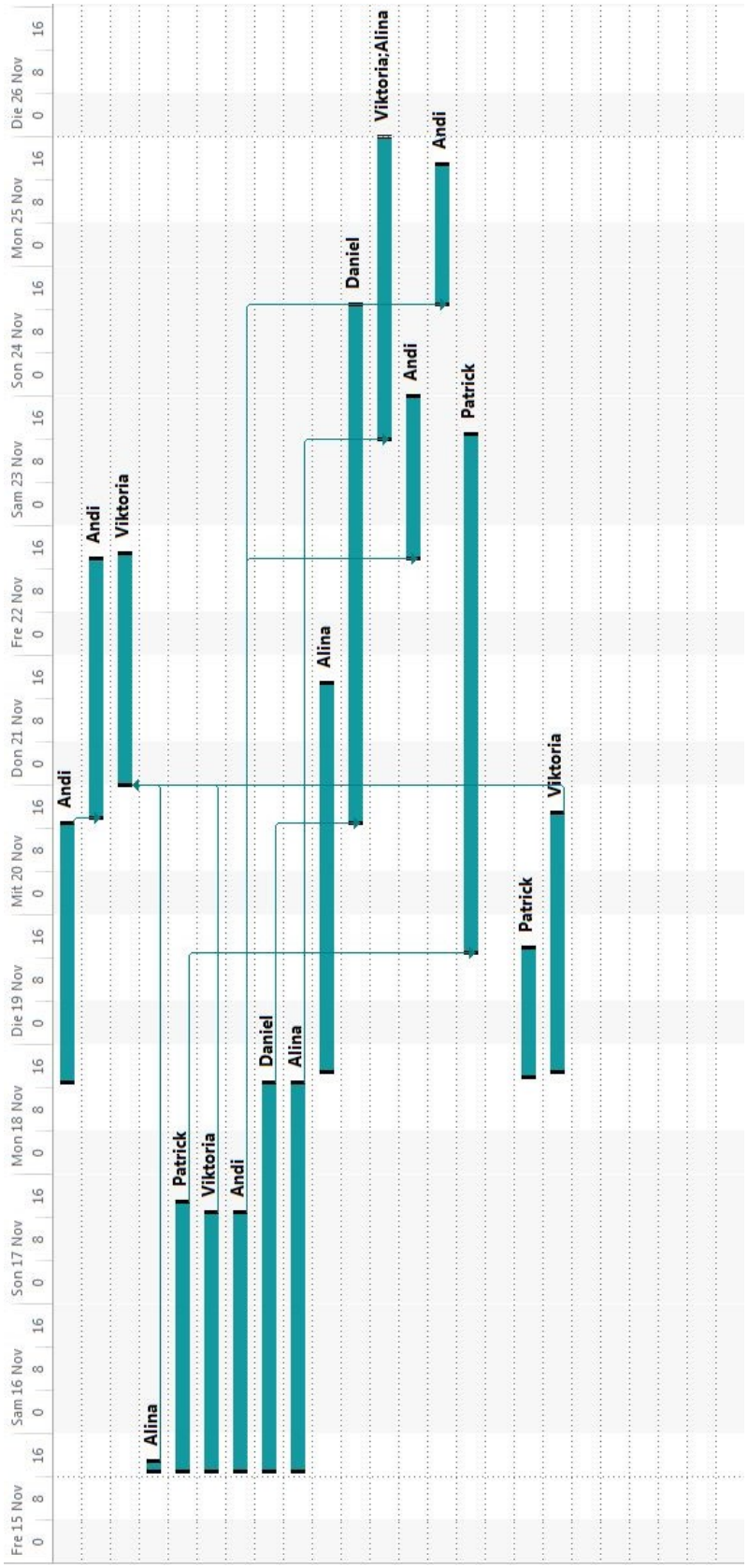
```
            assertTrue(lobby.playerSet.contains(player1));
            assertTrue(lobby.playerSet.contains(player2));
            assertTrue(lobby.playerSet.contains(player3));
            assertTrue(lobby.playerSet.contains(player4));
        }
}
```

5 Implementierungsplan

5.1 Arbeitspakete

	Vorgangsname	Dauer	Anfang	Ende	Vorgänger	Ressourcennamen
1	Komplette View	8 Std.	Mon 18.11.13	Mit 20.11.13		Andi
2	Controller	8 Std.	Mit 20.11.13	Fre 22.11.13	1	Andi
3	Server(Game+Lobby	8 Std.	Mit 20.11.13	Fre 22.11.13	4;6;18	Viktoria
4	ComOjbects	2 Std.	Fre 15.11.13	Fre 15.11.13		Alina
5	Testfälle Client	10 Std.	Fre 15.11.13	Son 17.11.13		Patrick
6	Testfälle Server	8 Std.	Fre 15.11.13	Son 17.11.13		Viktoria
7	Testfälle ClientRule	8 Std.	Fre 15.11.13	Son 17.11.13		Andi
8	Testfälle ServerRule	12 Std.	Fre 15.11.13	Mon 18.11.13		Daniel
9	Testfälle ServerRule	12 Std.	Fre 15.11.13	Mon 18.11.13		Alina
10	Ruleset-Daten	12 Std.	Mon 18.11.13	Don 21.11.13		Alina
11	WizardServerRulese	16 Std.	Mit 20.11.13	Son 24.11.13	8	Daniel
12	HeartsServerRulese	12 Std.	Sam 23.11.13	Mon 25.11.13	9	Viktoria;Alina
13	WizardClientRulese	6 Std.	Fre 22.11.13	Sam 23.11.13	7	Andi
14	HeartsClientRuleset	6 Std.	Son 24.11.13	Mon 25.11.13	7	Andi
15	ClientModel	16 Std.	Die 19.11.13	Sam 23.11.13	5	Patrick
16						
17	Client-Daten	4 Std.	Mon 18.11.13	Die 19.11.13		Patrick
18	Server(Player+Date	8 Std.	Mon 18.11.13	Mit 20.11.13		Viktoria

5.2 Gantt-Diagramm



Index