EasySet

Erzeugt von Doxygen 1.8.9.1

Mon Jan 12 2015 01:04:07

Inhaltsverzeichnis

Kapitel 1

Verzeichnis der Namensbereiche

1.1 Liste aller Namensbereiche

Liste aller Namensbereiche mit Kurzbeschreibung:

Kapitel 2

Hierarchie-Verzeichnis

2.1 Klassenhierarchie

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

Kapitel 3

Klassen-Verzeichnis

3.1 Auflistung der Klassen

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

Kapitel 4

Datei-Verzeichnis

4.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

Kapitel 5

Dokumentation der Namensbereiche

5.1 Ui-Namensbereichsreferenz

Kapitel 6

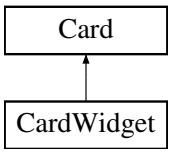
Klassen-Dokumentation

6.1 Card Klassenreferenz

Die Klasse Card stellt das digitale pendant zu den analogen Karten des Spiels Set dar.

`#include <card.hpp>`

Klassendiagramm für Card:



Öffentliche Methoden

- Card (short p_color, short p_shape, short p_number, short p_opacity)

  Konstruktor zum Erzeugen einer Card-Instanz.
- Card operator+ (Card &p_card)

  Überladener Plus-Operator, um Karten wie Vektoren zu addieren.
- char attributesToByte ()

  Konvertiert alle Attribute zu einem Byte.
- operator char ()

  Char-Konvertierungsoperator.

Geschützte Attribute

- short m_color

  Farbe der Karte (Wert im Intervall I = [0;2])
- short m_shape

  Form der Karte (Wert im Intervall I = [0;2])
- short m_number

  Anzahl der Karte (Wert im Intervall I = [0;2])
- short m_opacity

  Einfärbung der Karte (Wert im Intervall I = [0;2])

Freundbeziehungen

- class Window
- class Controller

6.1.1 Ausführliche Beschreibung

Die Klasse Card stellt das digitale pendant zu den analogen Karten des Spiels Set dar.

6.1.2 Beschreibung der Konstruktoren und Destruktoren

6.1.2.1 Card::Card (short *p_color,* short *p_shape,* short *p_number,* short *p_opacity*)

Konstruktor zum Erzeugen einer Card-Instanz.

Parameter

p_color	Farbe der zu erzeugenden Karte
p_shape	Form der zu erzeugenden Karte
p_number	Anzahl der zu erzeugenden Karte
p_opacity	Einfärbung der zu erzeugenden Karte

6.1.3 Dokumentation der Elementfunktionen

6.1.3.1 char Card::attributesToByte ( )

Konvertiert alle Attribute zu einem Byte.

Rückgabe

1 Byte, das alle Attribute umfasst

6.1.3.2 Card::operator char ( )

Char-Konvertierungsoperator.

6.1.3.3 Card Card::operator+ (Card & *p_card*)

Überladener Plus-Operator, um Karten wie Vektoren zu addieren.

Parameter

p_card	Karte die als 2. Summand dient

Rückgabe

Summe aus der aktuellen Karte und der übergebenen Karte

6.1.4 Freundbeziehungen und Funktionsdokumentation

6.1.4.1 friend class Controller [friend]

6.1.4.2 friend class Window [friend]

6.1.5 Dokumentation der Datenelemente

6.1.5.1 short Card::m_color [protected]

Farbe der Karte (Wert im Intervall I = [0;2])

Siehe auch

Color

6.1.5.2 short Card::m_number [protected]

Anzahl der Karte (Wert im Intervall I = [0;2])

Siehe auch

Number

6.1.5.3 short Card::m_opacity [protected]

Einfärbung der Karte (Wert im Intervall I = [0;2])

Siehe auch

Opacity

6.1.5.4 short Card::m_shape [protected]

Form der Karte (Wert im Intervall I = [0;2])

Siehe auch

Shape

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

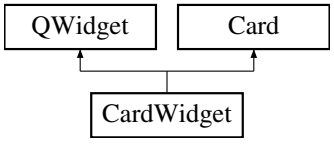
- src/card.hpp
- src/card.cpp

6.2 CardWidget Klassenreferenz

Die grafische Implementierung der Klasse Card.

`#include <cardwidget.hpp>`

Klassendiagramm für CardWidget:



Öffentliche Slots

- void unselect ()

  Wird aufgerufen, wenn eine Karte "entwählt" werden soll.
- void canClick (bool p_val)

  Wird aufgerufen, um festzulegen, ob man das CardWidgets anklicken kann oder eben nicht.

Signale

- void clicked ()

  Signal das emittiert wird, sobald ein Mausklick durchgeführt wird.

Öffentliche Methoden

- CardWidget (short p_color, short p_shape, short p_number, short p_opacity, QWidget ∗parent=0)

  Konstruktor um eine CardWidget-Instanz zu erzeugen.

Weitere Geerbte Elemente

6.2.1 Ausführliche Beschreibung

Die grafische Implementierung der Klasse Card.

Siehe auch

Card

6.2.2 Beschreibung der Konstruktoren und Destruktoren

6.2.2.1 CardWidget::CardWidget (short *p_color,* short *p_shape,* short *p_number,* short *p_opacity,* QWidget ∗ *parent* = `0`) `[explicit]`

Konstruktor um eine CardWidget-Instanz zu erzeugen.

Parameter

p_color	Farbe des CardWudgets
p_shape	Form des CardWidgets
p_number	Anzahl des CardWidgets
p_opacity	Einfärbung des CardWidgets
parent	Elternteil des CardWidgets

Siehe auch

Color
Shape
Number
Opacity

6.2.3 Dokumentation der Elementfunktionen

6.2.3.1 void CardWidget::canClick (bool *p_val*) `[slot]`

Wird aufgerufen, um festzulegen, ob man das CardWidgets anklicken kann oder eben nicht.

Parameter

p_val	Wahrheitswert der angibt, ob man das CardWidgets anklicken kann

6.2.3.2 void CardWidget::clicked ( ) [signal]

Signal das emittiert wird, sobald ein Mausklick durchgeführt wird.

6.2.3.3 void CardWidget::unselect ( ) [slot]

Wird aufgerufen, wenn eine Karte "entwählt" werden soll.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

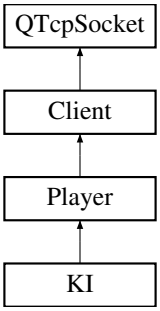
- src/cardwidget.hpp
- src/cardwidget.cpp

6.3 Client Klassenreferenz

Die abstrakte Klasse Client kümmert sich um die Verbindung zum Server und macht so ein Spielen möglich.

`#include <client.hpp>`

Klassendiagramm für Client:



Öffentliche Slots

- void onReadyRead ()

  Wird aufgerufen, sobald der Buffer mit den eingehenden Paketen lesebereit ist.

Öffentliche Methoden

- Client (QObject ∗p_parent=0, QHostAddress p_ip=QHostAddress::LocalHost, int p_port=1337)

  Konstruktor zum Erzeugen einer Client-Instanz.
- virtual void sendClickPacket (QByteArray p_cards)=0

  Sendet ein Klick-Paket - mit den ausgewählten Karten - an den Server.
  (Muss von Klasse Player definiert werden)
- virtual void sendTurnPacket ()=0

  Sendet ein Paket an den Server das angibt, dass der aktuelle Socket am Zug ist.
  (Muss von Klasse Player definiert werden)

Geschützte Attribute

- PacketHandler * m_packetHandler

  Paketverwalter für ein- und ausgehende Pakete.

6.3.1 Ausführliche Beschreibung

Die abstrakte Klasse Client kümmert sich um die Verbindung zum Server und macht so ein Spielen möglich.

6.3.2 Beschreibung der Konstruktoren und Destruktoren

6.3.2.1 Client::Client (QObject * *p_parent* = `0`, QHostAddress *p_ip* = `QHostAddress::LocalHost`, int *p_port* = `1337`) `[explicit]`

Konstruktor zum Erzeugen einer Client-Instanz.

Parameter

p_parent	Elternteil des Clients
p_ip	IP-Adresse mit der sich der Client verbinden soll
p_port	Port, auf dem die Verbindung stattfindet

6.3.3 Dokumentation der Elementfunktionen

6.3.3.1 void Client::onReadyRead ( ) `[slot]`

Wird aufgerufen, sobald der Buffer mit den eingehenden Paketen lesebereit ist.

6.3.3.2 virtual void Client::sendClickPacket (QByteArray *p_cards*) `[pure virtual]`

Sendet ein Klick-Paket - mit den ausgewählten Karten - an den Server.
(Muss von Klasse Player definiert werden)

Parameter

p_cards	Array bestehend aus den angewählten Karten

Implementiert in Player.

6.3.3.3 virtual void Client::sendTurnPacket ( ) `[pure virtual]`

Sendet ein Paket an den Server das angibt, dass der aktuelle Socket am Zug ist.
(Muss von Klasse Player definiert werden)

Implementiert in Player.

6.3.4 Dokumentation der Datenelemente

6.3.4.1 PacketHandler* Client::m_packetHandler `[protected]`

Paketverwalter für ein- und ausgehende Pakete.

Siehe auch

PacketHandler

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

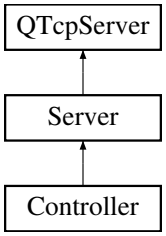
- src/client.hpp
- src/client.cpp

6.4 Controller Klassenreferenz

Die Klasse Controller ist für den kompletten Spielverlauf zuständig.
Sie regelt, wann welcher Client welche Pakete empfängt, wann das Spiel zu Ende ist/anfängt und wenn Karten nachgelegt werden sollen.

```
#include <controller.hpp>
```

Klassendiagramm für Controller:



Öffentliche Slots

- void draw (unsigned int p_count=3)

  Zieht eine bestimmte Anzahl zufälliger Karten aus dem Deck und legt sie aufs Spielfeld.
  Danach wird eine Spielfeld-Synchronisation ausgeführt.
- void retrieveClick (QTcpSocket ∗p_client, QByteArray p_cards)

  Wird aufgerufen, wenn ein Set ausgewählt wurde.
  Das Set wird überprüft und dementsprechend Punkte gutgeschrieben oder abgezogen.
- void retrievePlayerTurn (QTcpSocket ∗p_socket)

  Wird aufgerufen, sobald ein Spieler ein Set auswählen möchte.

Signale

- void showStartButton ()

  Wird emittiert, um den Startbutton auf dem Hauptfenster anzuzeigen.

Öffentliche Methoden

- void sendGameStartedPacket ()

  Sendet das Paket, um das Spiel zu starten.
- Controller (QObject ∗p_parent=0)

  Konstruktor zum Erzeugen einer Controller-Instanz.

Öffentliche, statische Methoden

- static bool check (Cards &p_cards)

  Überprüft ob eine Liste aus Karten ein Set bilden.

Weitere Geerbte Elemente

6.4.1 Ausführliche Beschreibung

Die Klasse Controller ist für den kompletten Spielverlauf zuständig.
Sie regelt, wann welcher Client welche Pakete empfängt, wann das Spiel zu Ende ist/anfängt und wenn Karten nachgelegt werden sollen.

6.4.2 Beschreibung der Konstruktoren und Destruktoren

6.4.2.1 Controller::Controller (QObject * *p_parent* = 0)

Konstruktor zum Erzeugen einer Controller-Instanz.

Parameter

p_parent	Elternteil des Controllers

6.4.3 Dokumentation der Elementfunktionen

6.4.3.1 bool Controller::check (Server::Cards & *p_cards*) [static]

Überprüft ob eine Liste aus Karten ein Set bilden.

Parameter

p_cards	Liste aus zu überprüfenden Karten

Rückgabe

Wahrheitswert der angibt, ob die Karten ein Set bilden

6.4.3.2 void Controller::draw (unsigned int *p_count* = 3) [slot]

Zieht eine bestimmte Anzahl zufälliger Karten aus dem Deck und legt sie aufs Spielfeld.
Danach wird eine Spielfeld-Synchronisation ausgeführt.

Parameter

p_count	

6.4.3.3 void Controller::retrieveClick (QTcpSocket * *p_client*, QByteArray *p_cards*) [slot]

Wird aufgerufen, wenn ein Set ausgewählt wurde.
Das Set wird überprüft und dementsprechend Punkte gutgeschrieben oder abgezogen.

Parameter

p_client	Spieler, der das Set ausgewählt hat
p_cards	Die ausgewählten Karten

Siehe auch

PacketHandler::readClick

6.4.3.4 void Controller::retrievePlayerTurn (QTcpSocket * *p_socket*) [slot]

Wird aufgerufen, sobald ein Spieler ein Set auswählen möchte.

Parameter

p_socket	Client, dem das Anklicken der Karten gewährt ist

Siehe auch

PacketHandler::readTurnPacket

6.4.3.5 void Controller::sendGameStartedPacket ( ) [virtual]

Sendet das Paket, um das Spiel zu starten.

Siehe auch

PacketHandler::makeGameStartedPacket

Implementiert Server.

6.4.3.6 void Controller::showStartButton ( ) [signal]

Wird emittiert, um den Startbutton auf dem Hauptfenster anzuzeigen.

Siehe auch

Window::retrieveShowStartButton

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

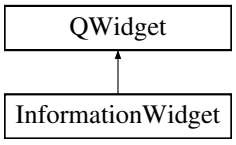
- src/controller.hpp
- src/controller.cpp

6.5 InformationWidget Klassenreferenz

Das InformationWidget dient zur Ausgabe des Spielstatuses.

`#include <informationwidget.hpp>`

Klassendiagramm für InformationWidget:



Öffentliche Methoden

- InformationWidget (QWidget *parent=0)
  Konstruktor zum Erzeugen einer InformationWidget-Instanz.
- void setPlayerCount (short p_playerCount)
  Legt die Anzahl der Spieler fest.
- void setDeckLength (short p_deckLength)
  Legt die Anzahl der Karten im Deck fest.
- void setScores (QByteArray p_scores)
  Legt den Inhalt des Scoreboards fest.

6.5.1 Ausführliche Beschreibung

Das InformationWidget dient zur Ausgabe des Spielstatuses.

6.5.2 Beschreibung der Konstruktoren und Destruktoren

6.5.2.1 InformationWidget::InformationWidget (QWidget * *parent* = 0) [explicit]

Konstruktor zum Erzeugen einer InformationWidget-Instanz.

Parameter

parent	Elternteil des InformationWidgets

6.5.3 Dokumentation der Elementfunktionen

6.5.3.1 void InformationWidget::setDeckLength (short *p_deckLength*)

Legt die Anzahl der Karten im Deck fest.

Parameter

p_deckLength	Anzahl der Karten im Deck

Siehe auch

m_deckLength

6.5.3.2 void InformationWidget::setPlayerCount (short *p_playerCount*)

Legt die Anzahl der Spieler fest.

Parameter

p_playerCount	Spieleranzahl

Siehe auch

m_playerCount

6.5.3.3 void InformationWidget::setScores (QByteArray *p_scores*)

Legt den Inhalt des Scoreboards fest.

Parameter

p_scores	Punktestand

Siehe auch

m_scores

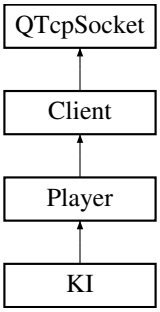
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- src/informationwidget.hpp
- src/informationwidget.cpp

6.6 KI Klassenreferenz

`#include <ki.hpp>`

Klassendiagramm für KI:



Öffentliche Slots

- void retrieveInputUnlocked ()
- void retrieveInputLocked ()
- void retrieveGameStarted ()
- void retrieveGameEnded ()
- void retrieveField (QByteArray p_field)

Öffentliche Methoden

- KI (QObject ∗p_parent=0, QHostAddress p_ip=QHostAddress::LocalHost, int p_port=1337)
- ∼KI ()

Weitere Geerbte Elemente

6.6.1 Beschreibung der Konstruktoren und Destruktoren

6.6.1.1 KI::KI (QObject ∗ *p_parent* = `0`, QHostAddress *p_ip* = `QHostAddress::LocalHost`, int *p_port* = `1337`)

6.6.1.2 KI::∼KI ( )

6.6.2 Dokumentation der Elementfunktionen

6.6.2.1 void KI::retrieveField (QByteArray *p_field*) `[slot]`

6.6.2.2 void KI::retrieveGameEnded ( ) `[slot]`

6.6.2.3 void KI::retrieveGameStarted ( ) `[slot]`

6.6.2.4 void KI::retrieveInputLocked ( ) `[slot]`

6.6.2.5 void KI::retrieveInputUnlocked ( ) `[slot]`

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

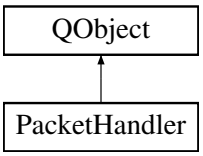
- src/ki.hpp
- src/ki.cpp

6.7 PacketHandler Klassenreferenz

Die PacketHandler-Klasse dient zur Verwaltung von ein- bzw. ausgehenden Paketen. Werden bestimmte Pakettypen erkannt, so werden die entsprechenden Signale emittiert, und dadurch die Slots in den externen Klassen aufgerufen.

`#include <packethandler.hpp>`

Klassendiagramm für PacketHandler:



Signale

- void readField (QByteArray p_field)

  Signal das emittiert wird, wenn das Spielfeldsynchronisations-Paket verarbeitet wird.
- void readScores (QByteArray p_scores)

  Signal das emittiert wird, wenn das Punktestandsynchronisations-Paket verarbeitet wird.
- void readClick (QTcpSocket ∗p_socket, QByteArray p_cards)

  Signal das emittiert wird, wenn ein Klick-Paket verarbeitet wird.
- void readGameStartedPacket ()

  Signal das emittiert wird, wenn das Spielstarten-Paket verarbeitet wird.
- void readGameFinishedPacket ()

  Signal das emittiert wird, wenn das Spielbeenden-Paket verarbeitet wird.
- void readDeckLength (short p_deckLength)

  Signal das emittiert wird, wenn das Deckgrößen-Paket verarbeitet wird.
- void readLockedPacket ()

  Signal das emittiert wird, wenn das Eingabensperr-Paket verarbeitet wird.
- void readUnlockedPacket ()

  Signal das emittiert wird, wenn das Eingabenentsperr-Paket verarbeitet wird.
- void readTurnPacket (QTcpSocket ∗p_socket)

  Signal das emittiert wird, wenn das Spieler-Am-Zug-Paket verarbeitet wird.

Öffentliche Methoden

- PacketHandler (QObject ∗parent=0)

  Konstruktor zum Erzeugen einer PacketHandler-Instanz.
- void processPackets (QByteArray p_packets, QTcpSocket ∗p_socket=nullptr)

  Teilt den Datenstrom in die einzelnen Pakete auf und verarbeitet diese dann.
- QByteArray makeFSPacket (std::list< Card ∗ > &p_field)

  Erzeugt ein Paket für die Spielfeldsynchronisation, anhand einer Liste aus Karte.
- QByteArray makeClickPacket (QByteArray p_cards)

  Erzeugt ein Paket das eine Set-Auswahl darstellt.
- QByteArray makeScoresPacket (QByteArray p_scores)

  Erzeugt ein Paket, für die Punkte-Synchronisation.
- QByteArray makeGameStartedPacket ()

  Erzeugt eine Paket das den Start des Spiels darstellt.
- QByteArray makeGameFinishedPacket ()

Erzeugt ein Paket, um das Spielende anzusagen.

- QByteArray makeDeckLengthPacket (short p_deckLength)

  Erzeugt ein Paket das die Anzahl der im Deck beinhalteten Karten darstellt.
- QByteArray makeInputLockedPacket ()

  Erzeugt ein Paket das die Sperrung der Eingaben bewirkt.
- QByteArray makeInputUnlockedPacket ()

  Erzeugt ein Paket das die Entsperrung der Eingaben bewirkt.
- QByteArray makeTurnPacket ()

  Erzeugt ein Paket das angibt, dass ein Spieler ein Set auswählen möchte.

6.7.1 Ausführliche Beschreibung

Die PacketHandler-Klasse dient zur Verwaltung von ein- bzw. ausgehenden Paketen.
Werden bestimmte Pakettypen erkannt, so werden die entsprechenden Signale emittiert,
und dadurch die Slots in den externen Klassen aufgerufen.

6.7.2 Beschreibung der Konstruktoren und Destruktoren

6.7.2.1 PacketHandler::PacketHandler (QObject * *parent* = 0) [explicit]

Konstruktor zum Erzeugen einer PacketHandler-Instanz.

Parameter

parent	Elternteil des PacketHandlers

6.7.3 Dokumentation der Elementfunktionen

6.7.3.1 QByteArray PacketHandler::makeClickPacket (QByteArray *p_cards*)

Erzeugt ein Paket das eine Set-Auswahl darstellt.

Parameter

p_cards	Ausgewählte Karten

Rückgabe

Paket

6.7.3.2 QByteArray PacketHandler::makeDeckLengthPacket (short *p_deckLength*)

Erzeugt ein Paket das die Anzahl der im Deck beinhalteten Karten darstellt.

Parameter

p_deckLength	Deckgröße

Rückgabe

Paket

6.7.3.3 QByteArray PacketHandler::makeFSPacket (std::list< Card * > & *p_field*)

Erzeugt ein Paket für die Spielfeldsynchronisation, anhand einer Liste aus Karte.

Parameter

p_field	Liste aus Karten - Spielfeld

Rückgabe

Paket

6.7.3.4 QByteArray PacketHandler::makeGameFinishedPacket ( )

Erzeugt ein Paket, um das Spielende anzusagen.

Rückgabe

Paket

6.7.3.5 QByteArray PacketHandler::makeGameStartedPacket ( )

Erzeugt eine Paket das den Start des Spiels darstellt.

Rückgabe

Paket

6.7.3.6 QByteArray PacketHandler::makeInputLockedPacket ( )

Erzeugt ein Paket das die Sperrung der Eingaben bewirkt.

Rückgabe

Paket

6.7.3.7 QByteArray PacketHandler::makeInputUnlockedPacket ( )

Erzeugt ein Paket das die Entsperrung der Eingaben bewirkt.

Rückgabe

Paket

6.7.3.8 QByteArray PacketHandler::makeScoresPacket (QByteArray *p_scores*)

Erzeugt ein Paket, für die Punkte-Synchronisation.

Parameter

p_scores	Die Punkte aller Spieler

Rückgabe

Paket

6.7.3.9 QByteArray PacketHandler::makeTurnPacket ( )

Erzeugt ein Paket das angibt, dass ein Spieler ein Set auswählen möchte.

Rückgabe

Paket

6.7.3.10 void PacketHandler::processPackets (QByteArray *p_packets,* QTcpSocket * *p_socket* = `nullptr`)

Teilt den Datenstrom in die einzelnen Pakete auf und verarbeitet diese dann.

Parameter

p_packets	Datenstrom
p_socket	Absender

Siehe auch

processPacket

6.7.3.11 void PacketHandler::readClick (QTcpSocket * *p_socket,* QByteArray *p_cards*) [signal]

Signal das emittiert wird, wenn ein Klick-Paket verarbeitet wird.

Parameter

p_socket	Absender
p_cards	Ausgewählte Karten

6.7.3.12 void PacketHandler::readDeckLength (short *p_deckLength*) [signal]

Signal das emittiert wird, wenn das Deckgrößen-Paket verarbeitet wird.

Parameter

p_deckLength	Deckgröße

6.7.3.13 void PacketHandler::readField (QByteArray *p_field*) [signal]

Signal das emittiert wird, wenn das Spielfeldsynchronisations-Paket verarbeitet wird.

Parameter

p_field	Spielfeld

6.7.3.14 void PacketHandler::readGameFinishedPacket ( ) [signal]

Signal das emittiert wird, wenn das Spielbeenden-Paket verarbeitet wird.

6.7.3.15 void PacketHandler::readGameStartedPacket ( ) [signal]

Signal das emittiert wird, wenn das Spielstarten-Paket verarbeitet wird.

6.7.3.16 void PacketHandler::readLockedPacket ( ) [signal]

Signal das emittiert wird, wenn das Eingabensperr-Paket verarbeitet wird.

6.7.3.17 void PacketHandler::readScores (QByteArray *p_scores*) [signal]

Signal das emittiert wird, wenn das Punktestandsynchronisations-Paket verarbeitet wird.

Parameter

p_scores	Punktestand

6.7.3.18 void PacketHandler::readTurnPacket (QTcpSocket * *p_socket*) [signal]

Signal das emittiert wird, wenn das Spieler-Am-Zug-Paket verarbeitet wird.

Parameter

p_socket	Spieler, der am Zug ist

6.7.3.19 void PacketHandler::readUnlockedPacket ( ) [signal]

Signal das emittiert wird, wenn das Eingabenentsperr-Paket verarbeitet wird.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

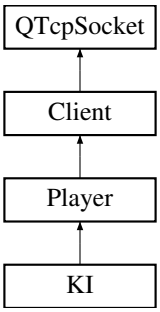
- src/packethandler.hpp
- src/packethandler.cpp

6.8 Player Klassenreferenz

Spieler-Klasse.

`#include <player.hpp>`

Klassendiagramm für Player:



Öffentliche Methoden

- void sendClickPacket (QByteArray p_cards)

  Sendet ein Paket an den Server, in dem die ausgewählten Karten drinnenstehen.
  Der Server überprüft diese auf ein Set und erhöht und senkt dementsprechend die Punktzahl des Spielers.
- void sendTurnPacket ()

Sendet ein Paket an den Server, dass dafür steht, dass dieser Spieler eine Set-Auswahl trifft.

- Player (QObject *p_parent=0, QHostAddress p_ip=QHostAddress::LocalHost, int p_port=1337)

  Konstruktor zum Erzeugen einer Player-Klasse.

Weitere Geerbte Elemente

6.8.1 Ausführliche Beschreibung

Spieler-Klasse.

6.8.2 Beschreibung der Konstruktoren und Destruktoren

6.8.2.1 Player::Player (QObject * *p_parent* = `0`, QHostAddress *p_ip* = `QHostAddress::LocalHost`, int *p_port* = `1337`)

Konstruktor zum Erzeugen einer Player-Klasse.

Parameter

p_parent	Elternteil des Players
p_ip	IP-Adresse des Servers
p_port	Port über den die Verbindung läuft

6.8.3 Dokumentation der Elementfunktionen

6.8.3.1 void Player::sendClickPacket (QByteArray *p_cards*) `[virtual]`

Sendet ein Paket an den Server, in dem die ausgewählten Karten drinnenstehen.
Der Server überprüft diese auf ein Set und erhöht und senkt dementsprechend die Punktzahl des Spielers.

Parameter

p_cards	Angeklickte Karten

Siehe auch

PacketHandler::makeClickPacket

Implementiert Client.

6.8.3.2 void Player::sendTurnPacket ( ) `[virtual]`

Sendet ein Paket an den Server, dass dafür steht, dass dieser Spieler eine Set-Auswahl trifft.

Implementiert Client.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

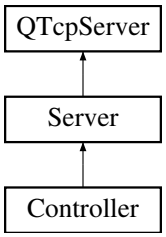
- src/player.hpp
- src/player.cpp

6.9 Server Klassenreferenz

Die abstrakte Klasse Server, steht für den Server des Spiels.

`#include <server.hpp>`

Klassendiagramm für Server:



Öffentliche Slots

- void newCon ()

  Wird aufgerufen, wenn eine neue Verbindung zur Verfügung steht.
- void onReadyRead ()

  Wird aufgerufen, sobald der Buffer mit den eingehenden Paketen lesebereit ist.
- virtual void retrieveClick (QTcpSocket ∗p_client, QByteArray p_cards)=0

  Wird aufgerufen, wenn ein Set ausgewählt wurde.
  Das Set wird überprüft und dementsprechend Punkte gutgeschrieben oder abgezogen.
- virtual void retrievePlayerTurn (QTcpSocket ∗p_socket)=0

  Wird aufgerufen, sobald ein Spieler ein Set auswählen möchte.

Öffentliche Methoden

- virtual void sendGameStartedPacket ()=0

  Sendet das Paket, um das Spiel zu starten.
- Server (QObject ∗parent=0, int p_port=1337)

  Konstruktor zum Erzeugen einer Server-Instanz.

Geschützte Typen

- typedef std::list< Card ∗ > Cards

  Typdefinition Cards.
- typedef std::tuple< QTcpSocket ∗, short, Cards > Client

  Typdefinition Client.
- typedef std::vector< Client > Clients

  Typdefinition Clients.

Geschützte Methoden

- virtual void sendFSPacket ()=0

  Sendet das Paket für die Spielfeldsynchronisation.
- virtual void sendDeckLengthPacket ()=0

  Sendet die Anzahl der verbleibenden Karten im Deck.
- virtual void sendScoreboard ()=0

  Sendet das Paket für die Punktzahl-Synchronisation.
- virtual void sendGameFinishedPacket ()=0

  Sendet das Paket, das angibt, dass das Spiel vorrüber ist und
  trennt die Verbindungen zu allen Clients.

- virtual void sendInputLocked ()=0

  Sendet das Paket, um die Eingaben der Spieler zu sperren, sobald ein Spieler ein Set auswählen möchte.
- virtual void sendInputUnlocked (QTcpSocket *p_socket=nullptr)=0

  Sendet das Paket, um die Eingaben eines bestimmten Spielers oder aller Spieler zu entsperren.

Geschützte Attribute

- Clients m_clients

  Liste aus allen verbundenen Clients.
- PacketHandler * m_packetHandler

  Paketverwalter für ein- und ausgehende Pakete.

6.9.1 Ausführliche Beschreibung

Die abstrakte Klasse Server, steht für den Server des Spiels.

6.9.2 Dokumentation der benutzerdefinierten Datentypen

6.9.2.1 typedef std::list<Card*> Server::Cards [protected]

Typdefinition Cards.

Siehe auch

Card

6.9.2.2 typedef std::tuple<QTcpSocket*, short, Cards> Server::Client [protected]

Typdefinition Client.

Siehe auch

Server::Cards

6.9.2.3 typedef std::vector<Client> Server::Clients [protected]

Typdefinition Clients.

Siehe auch

Server::Client

6.9.3 Beschreibung der Konstruktoren und Destruktoren

6.9.3.1 Server::Server (QObject * *parent* = 0, int *p_port* = 1337) [explicit]

Konstruktor zum Erzeugen einer Server-Instanz.

Parameter

parent	Elternteil des Servers
p_port	Port an dem der Server lauscht

6.9.4 Dokumentation der Elementfunktionen

6.9.4.1 void Server::newCon ( ) [slot]

Wird aufgerufen, wenn eine neue Verbindung zur Verfügung steht.

6.9.4.2 void Server::onReadyRead ( ) [slot]

Wird aufgerufen, sobald der Buffer mit den eingehenden Paketen lesebereit ist.

6.9.4.3 virtual void Server::retrieveClick (QTcpSocket * *p_client,* QByteArray *p_cards*) [pure virtual],[slot]

Wird aufgerufen, wenn ein Set ausgewählt wurde.
Das Set wird überprüft und dementsprechend Punkte gutgeschrieben oder abgezogen.

Parameter

p_client	Spieler, der das Set ausgewählt hat
p_cards	Die ausgewählten Karten

Siehe auch

PacketHandler::readClick

6.9.4.4 virtual void Server::retrievePlayerTurn (QTcpSocket * *p_socket*) [pure virtual],[slot]

Wird aufgerufen, sobald ein Spieler ein Set auswählen möchte.

Parameter

p_socket	Client, dem das Anklicken der Karten gewährt ist

Siehe auch

PacketHandler::readTurnPacket

6.9.4.5 virtual void Server::sendDeckLengthPacket ( ) [protected],[pure virtual]

Sendet die Anzahl der verbleibenden Karten im Deck.

Siehe auch

PacketHandler::makeDeckLengthPacket

6.9.4.6 virtual void Server::sendFSPacket ( ) [protected],[pure virtual]

Sendet das Paket für die Spielfeldsynchronisation.

Siehe auch

PacketHandler::makeFSPacket

6.9.4.7 virtual void Server::sendGameFinishedPacket ( ) [protected], [pure virtual]

Sendet das Paket, das angibt, dass das Spiel vorrüber ist und
trennt die Verbindungen zu allen Clients.

Siehe auch

PacketHandler::makeGameFinishedPacket

6.9.4.8 virtual void Server::sendGameStartedPacket ( ) [pure virtual]

Sendet das Paket, um das Spiel zu starten.

Siehe auch

PacketHandler::makeGameStartedPacket

Implementiert in Controller.

6.9.4.9 virtual void Server::sendInputLocked ( ) [protected], [pure virtual]

Sendet das Paket, um die Eingaben der Spieler zu sperren,
sobald ein Spieler ein Set auswählen möchte.

Siehe auch

PacketHandler::makeInputLockedPacket

6.9.4.10 virtual void Server::sendInputUnlocked (QTcpSocket * *p_socket* = nullptr) [protected], [pure virtual]

Sendet das Paket, um die Eingaben eines bestimmten Spielers
oder aller Spieler zu entsperren.

Parameter

p_socket	

Siehe auch

PacketHandler::makeInputUnlockedPacket

6.9.4.11 virtual void Server::sendScoreboard ( ) [protected], [pure virtual]

Sendet das Paket für die Punktzahl-Synchronisation.

Siehe auch

PacketHandler::makeScoresPacket

6.9.5 Dokumentation der Datenelemente

6.9.5.1 Clients Server::m_clients [protected]

Liste aus allen verbundenen Clients.

Siehe auch

Server::Clients

6.9.5.2 PacketHandler* Server::m_packetHandler [protected]

Paketverwalter für ein- und ausgehende Pakete.

Siehe auch

PacketHandler

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

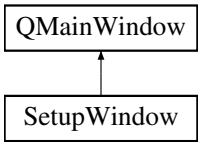
- src/server.hpp
- src/server.cpp

6.10 SetupWindow Klassenreferenz

Die Klasse SetupWindow ist ein kleiner Einrichtungsassistent, um ein Spiel zu konfigurieren.

`#include <setupwindow.hpp>`

Klassendiagramm für SetupWindow:



Öffentliche Methoden

- SetupWindow (QWidget *parent=0)

  Konstruktor zum Erzeugen einer SetupWindow-Instanz.
- ~SetupWindow ()

  Destruktor zum Aufräumen einer SetupWindow-Instanz.

6.10.1 Ausführliche Beschreibung

Die Klasse SetupWindow ist ein kleiner Einrichtungsassistent, um ein Spiel zu konfigurieren.

6.10.2 Beschreibung der Konstruktoren und Destruktoren

6.10.2.1 SetupWindow::SetupWindow (QWidget * *parent* = 0) [explicit]

Konstruktor zum Erzeugen einer SetupWindow-Instanz.

Parameter

parent	Elternteil des SetupWindows

6.10.2.2 SetupWindow::~SetupWindow ( )

Destruktor zum Aufräumen einer SetupWindow-Instanz.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

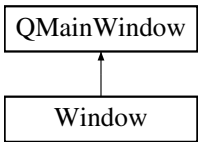
- src/setupwindow.hpp
- src/setupwindow.cpp

6.11 Window Klassenreferenz

Die Singleton-Klasse Window ist die HauptGUI.
Dort findet das Spiel so wirklich für die Spieler statt.

```
#include <window.hpp>
```

Klassendiagramm für Window:



Öffentliche Slots

- void cardClicked ()

  Wird aufgerufen, wenn eine Karte angeklickt wird.
- void clientDisconnected ()

  Wird aufgerufen, wenn die Verbindung zum Server unterbrochen/ getrennt wird.
- void retrieveField (QByteArray p_field)

  Wird aufgerufen, wenn das Spielfeldsynchronisations-Paket empfangen wird.
- void retrieveDeckLength (short p_deckLength)

  Wird aufgerufen, wenn das Deckgröße-Paket empfangen wird.
- void retrieveScores (QByteArray p_scores)

  Wird aufgerufen, wenn das Punktestand-Paket empfangen wird.
- void retrieveGameStarted ()

  Wird aufgerufen, wenn das Spielstarten-Paket empfangen wird.
- void retrieveGameFinished ()

  Wird aufgerufen, wenn das Spielbeenden-Paket empfangen wird.
- void retrieveShowStartButton ()

  Wird aufgerufen, wenn der Start-Button angezeigt werden soll.
- void retrieveUnlock ()

  Wird aufgerufen, wenn das Eingabenentsperr-Paket emfpangen wird.
- void retrieveLock ()

  Wird aufgerufen, wenn das Eingabesperr-Paket empfangen wird.

Signale

- void unselectAll ()

  Signal das alle Karten entwählt.
- void canClick (bool p_val)

  Signal das den Status der Karten - ob man diese anklicken kann oder nicht - festlegt.

Öffentliche, statische Methoden

- static Window & getInstance ()

  Liefert die Instanz der Singleton-Klasse Window.

Öffentliche Attribute

- std::list< std::tuple< Player *, Qt::Key > > m_players

  Liste aus allen aktuellen, lokalen Spielern und deren ausgewählte Taste.

Geschützte Methoden

- void closeEvent (QCloseEvent *p_closeEvent)

  Wird aufgerufen, wenn der Schließen-Button des Fensters geklickt wird.
- void keyPressEvent (QKeyEvent *p_keyEvent)

  Wird aufgerufen, wenn eine Taste auf der Tastatur gedrückt wird.

6.11.1 Ausführliche Beschreibung

Die Singleton-Klasse Window ist die HauptGUI.
Dort findet das Spiel so wirklich für die Spieler statt.

6.11.2 Dokumentation der Elementfunktionen

6.11.2.1 void Window::canClick (bool *p_val*) [signal]

Signal das den Status der Karten - ob man diese anklicken kann oder nicht - festlegt.

Parameter

p_val	Wahrheitswert der angibt, ob die Karten anklickbar sind oder nicht

6.11.2.2 void Window::cardClicked ( ) [slot]

Wird aufgerufen, wenn eine Karte angeklickt wird.

Siehe auch

CardWidget::clicked

6.11.2.3 void Window::clientDisconnected ( ) [slot]

Wird aufgerufen, wenn die Verbindung zum Server unterbrochen/ getrennt wird.

6.11.2.4 void Window::closeEvent (QCloseEvent * *p_closeEvent*) [protected]

Wird aufgerufen, wenn der Schließen-Button des Fensters geklickt wird.

Parameter

p_closeEvent	Schließ-Event

6.11.2.5 Window & Window::getInstance ( ) [static]

Liefert die Instanz der Singleton-Klasse Window.

Rückgabe

Window-Instanz

6.11.2.6 void Window::keyPressEvent (QKeyEvent * *p_keyEvent*) [protected]

Wird aufgerufen, wenn eine Taste auf der Tastatur gedrückt wird.

Parameter

p_keyEvent	Tasten-Event

6.11.2.7 void Window::retrieveDeckLength (short *p_deckLength*) [slot]

Wird aufgerufen, wenn das Deckgröße-Paket empfangen wird.

Parameter

p_deckLength	Deckgröße

Siehe auch

PacketHandler::readDeckLength

6.11.2.8 void Window::retrieveField (QByteArray *p_field*) [slot]

Wird aufgerufen, wenn das Spielfeldsynchronisations-Paket empfangen wird.

Parameter

p_field	Spielfeld

Siehe auch

PacketHandler::readField

6.11.2.9 void Window::retrieveGameFinished ( ) [slot]

Wird aufgerufen, wenn das Spielbeenden-Paket empfangen wird.

Siehe auch

PacketHandler::readGameFinishedPacket

6.11.2.10 void Window::retrieveGameStarted ( ) [slot]

Wird aufgerufen, wenn das Spielstarten-Paket empfangen wird.

Siehe auch

PacketHandler::readGameStartedPacket

6.11.2.11 void Window::retrieveLock ( ) [slot]

Wird aufgerufen, wenn das Eingabesperr-Paket empfangen wird.

Siehe auch

PacketHandler::readLockedPacket

6.11.2.12 void Window::retrieveScores (QByteArray *p_scores*) [slot]

Wird aufgerufen, wenn das Punktestand-Paket empfangen wird.

Parameter

p_scores	Punktestand

Siehe auch

PacketHandler::readScores

6.11.2.13 void Window::retrieveShowStartButton ( ) [slot]

Wird aufgerufen, wenn der Start-Button angezeigt werden soll.

Siehe auch

Controller::showStartButton

6.11.2.14 void Window::retrieveUnlock ( ) [slot]

Wird aufgerufen, wenn das Eingabenentsperr-Paket emfpangen wird.

Siehe auch

PacketHandler::readUnlockedPacket

6.11.2.15 void Window::unselectAll ( ) [signal]

Signal das alle Karten entwählt.

6.11.3 Dokumentation der Datenelemente

6.11.3.1 std::list<std::tuple<Player*, Qt::Key> > Window::m_players

Liste aus allen aktuellen, lokalen Spielern und deren ausgewählte Taste.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- src/window.hpp
- src/window.cpp

Kapitel 7

Datei-Dokumentation

7.1 src/card.cpp-Dateireferenz

```
#include "card.hpp"
```

7.2 src/card.hpp-Dateireferenz

```
#include "enums.hpp"
#include "cardwidget.hpp"
#include "window.hpp"
#include "controller.hpp"
```

Klassen

- class Card

  Die Klasse Card stellt das digitale pendant zu den analogen Karten des Spiels Set dar.

7.3 src/cardwidget.cpp-Dateireferenz

```
#include "cardwidget.hpp"
```

7.4 src/cardwidget.hpp-Dateireferenz

```
#include <atomic>
#include <QWidget>
#include <QPainter>
#include <QPaintEvent>
#include <QMouseEvent>
#include "card.hpp"
```

Klassen

- class CardWidget

  Die grafische Implementierung der Klasse Card.

7.5 src/client.cpp-Dateireferenz

```
#include "client.hpp"
```

7.6 src/client.hpp-Dateireferenz

```
#include <QTcpSocket>
#include <QHostAddress>
#include "packethandler.hpp"
```

Klassen

- class Client

  Die abstrakte Klasse Client kümmert sich um die Verbindung zum Server und macht so ein Spielen möglich.

7.7 src/controller.cpp-Dateireferenz

```
#include "controller.hpp"
```

7.8 src/controller.hpp-Dateireferenz

```
#include <atomic>
#include <list>
#include <stdlib.h>
#include <ctime>
#include <thread>
#include <chrono>
#include <QTimer>
#include "server.hpp"
#include "card.hpp"
```

Klassen

- class Controller

  Die Klasse Controller ist für den kompletten Spielverlauf zuständig.
  Sie regelt, wann welcher Client welche Pakete empfängt, wann das Spiel zu Ende ist/anfängt und wenn Karten nachgelegt werden sollen.

7.9 src/enums.hpp-Dateireferenz

```
#include <QColor>
```

Aufzählungen

- enum PacketHeader {
  GAME_STATE = 0x01, DECK = 0x02, INPUT_STATE = 0x03, PLAYER_TURN = 0x04,
  SCORES = 0x05, FIELD_SYNCHRO = 0x07, CLICK = 0x0A }

  Anonyme innere Klasse, mit den verschiedenen Pakettypen.
- enum Color { RED, BLUE, GREEN }

  Anonyme innere Klasse, mit den verschiedenen Farben einer Karte.
- enum Shape { TRIANGLE, SQUARE, CIRCLE }

  Anonyme innere Klasse, mit den verschiedenen Formen einer Karte.
- enum Number { ONE, TWO, THREE }

  Anonyme innere Klasse, mit den verschiedenen Anzahlen einer Karte.
- enum Opacity { FILLED, EMPTY, PALLID }

  Anonyme innere Klasse, mit den verschiedenen Einfärbungen einer Karte.

7.9.1 Dokumentation der Aufzählungstypen

7.9.1.1 enum Color

Anonyme innere Klasse, mit den verschiedenen Farben einer Karte.

Aufzählungswerte

RED

BLUE

GREEN

7.9.1.2 enum Number

Anonyme innere Klasse, mit den verschiedenen Anzahlen einer Karte.

Aufzählungswerte

ONE

TWO

THREE

7.9.1.3 enum Opacity

Anonyme innere Klasse, mit den verschiedenen Einfärbungen einer Karte.

Aufzählungswerte

FILLED

EMPTY

PALLID

7.9.1.4 enum PacketHeader

Anonyme innere Klasse, mit den verschiedenen Pakettypen.

Aufzählungswerte

GAME_STATE

DECK

INPUT_STATE

PLAYER_TURN

SCORES

FIELD_SYNCHRO

CLICK

7.9.1.5 enum Shape

Anonyme innere Klasse, mit den verschiedenen Formen einer Karte.

Aufzählungswerte

TRIANGLE

SQUARE

CIRCLE

7.10 src/informationwidget.cpp-Dateireferenz

```
#include "informationwidget.hpp"
```

7.11 src/informationwidget.hpp-Dateireferenz

```
#include <iostream>
#include <QWidget>
#include <QLabel>
```

Klassen

- class InformationWidget

  Das InformationWidget dient zur Ausgabe des Spielstatuses.

7.12 src/ki.cpp-Dateireferenz

```
#include "ki.hpp"
```

7.13 src/ki.hpp-Dateireferenz

```
#include <atomic>
#include <thread>
#include <mutex>
#include <chrono>
#include "player.hpp"
#include "card.hpp"
```

Klassen

- class KI

7.14 src/main.cpp-Dateireferenz

```
#include <QApplication>
#include "setupwindow.hpp"
```

Funktionen

- int main (int argc, char ∗argv[ ])

7.14.1 Dokumentation der Funktionen

7.14.1.1 int main (int *argc,* char ∗ *argv[ ]*)

7.15 src/packethandler.cpp-Dateireferenz

```
#include "packethandler.hpp"
```

7.16 src/packethandler.hpp-Dateireferenz

```
#include <iostream>
#include <QObject>
#include <QTcpSocket>
#include "enums.hpp"
#include "card.hpp"
```

Klassen

- class PacketHandler

  Die PacketHandler-Klasse dient zur Verwaltung von ein- bzw. ausgehenden Paketen. Werden bestimmte Pakettypen erkannt, so werden die entsprechenden Signale emittiert, und dadurch die Slots in den externen Klassen aufgerufen.

7.17 src/player.cpp-Dateireferenz

```
#include "player.hpp"
```

7.18 src/player.hpp-Dateireferenz

```
#include <QObject>
#include <QByteArray>
#include "client.hpp"
```

Klassen

- class Player

  Spieler-Klasse.

7.19 src/server.cpp-Dateireferenz

```
#include "server.hpp"
```

7.20 src/server.hpp-Dateireferenz

```
#include <vector>
#include <tuple>
#include <QTcpServer>
#include "packethandler.hpp"
```

Klassen

- class Server

  Die abstrakte Klasse Server, steht für den Server des Spiels.

7.21 src/setupwindow.cpp-Dateireferenz

```
#include "src/setupwindow.hpp"
#include "ui_setupwindow.h"
```

7.22 src/setupwindow.hpp-Dateireferenz

```
#include <iostream>
#include <QMainWindow>
#include <QListWidgetItem>
#include <QKeyEvent>
#include <QDesktopWidget>
#include "window.hpp"
#include "player.hpp"
#include "ki.hpp"
#include "controller.hpp"
```

Klassen

- class SetupWindow

  Die Klasse SetupWindow ist ein kleiner Einrichtungsassistent, um ein Spiel zu konfigurieren.

Namensbereiche

- Ui

7.23 src/window.cpp-Dateireferenz

```
#include "window.hpp"
#include "ui_window.h"
```

7.24 src/window.hpp-Dateireferenz

```
#include <atomic>
#include <list>
#include <QDesktopWidget>
#include <QMainWindow>
#include <QWidget>
#include <QPushButton>
#include <QMessageBox>
#include <QKeyEvent>
#include "card.hpp"
#include "informationwidget.hpp"
#include "player.hpp"
#include "controller.hpp"
```

Klassen

- class Window

  Die Singleton-Klasse Window ist die HauptGUI.
  Dort findet das Spiel so wirklich für die Spieler statt.

Namensbereiche

- Ui

Index