# RaetselLoeser

Erzeugt von Doxygen 1.8.6

Fre Mai 13 2016 10:38:36

# Inhaltsverzeichnis

# Kapitel 1

# Verzeichnis der Namensbereiche

## 1.1 Pakete

Hier folgen die Pakete mit einer Kurzbeschreibung (wenn verfügbar):

# 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

# Dokumentation der Namensbereiche

## 4.1 Paket raetselErsteller

### Pakete

- package daten
- package io
- package logik

### Klassen

- class Konstanten

### 4.1.1 Ausführliche Beschreibung

Dieses Packet dient als uebergeordnetes Basisverzeichnis für alle Klassen dieses Projekts.

**Autor**

Felix Kibellus

## 4.2 Paket raetselErsteller.daten

### Klassen

- class AktuelleKombination
- class AusgabeDaten
- class EingabeDaten
- class Punkt
- class SchlechtesteAktuelleKombination

### 4.2.1 Ausführliche Beschreibung

In diesem Paket befinden sich alle Klassen, welche zur Datenschicht gehören.

**Autor**

Felix Kibellus

## 4.3 Paket raetselErsteller.io

### Pakete

- package format

### Klassen

- interface AusgabeDatenSchreiber
- class ConsoleErrorHandler
- interface EingabeDatenLeser
- interface ErrorHandler
- class FileIO
- class FileIOException

### 4.3.1 Ausführliche Beschreibung

In diesem Paket befinden sich alle Klassen, welche für Input und Output verwendet werden.

**Autor**

Felix Kibellus

## 4.4 Paket raetselErsteller.io.format

### Klassen

- class FormatException
- interface Formatierer
- class StdFormat

### 4.4.1 Ausführliche Beschreibung

In diesem Paket befinden sich alle Klassen, welche zur Formatierung von Dateien benötigt werden.

**Autor**

Felix Kibellus

## 4.5 Paket raetselErsteller.logik

### Pakete

- package algorithmus

### Klassen

- class Controller
- class Main

### 4.5.1 Ausführliche Beschreibung

Alle Klassen welche relevante Programmlogik oder Klassen mit Controlling-Aufgaben enthalten befinden sich in diesem Paket.

**Autor**

Felix Kibellus

## 4.6 Paket raetselErsteller.logik.algorithmus

### Klassen

- class BacktrackingAlgorithmus
- class KeineLoesungException
- interface LoesungsAlgorithmus

### 4.6.1 Ausführliche Beschreibung

In diesem Paket finden sich alle Klassen zum Kernalgorithmus.

**Autor**

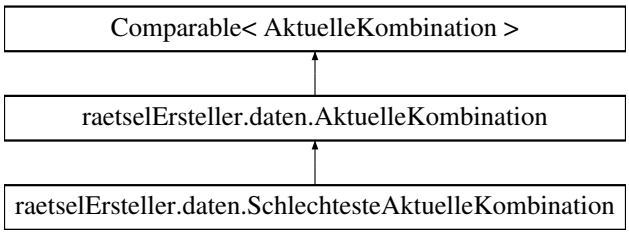
Felix Kibellus

# Kapitel 5

# Klassen-Dokumentation

## 5.1 raetselErsteller.daten.AktuelleKombination Klassenreferenz

Klassendiagramm für raetselErsteller.daten.AktuelleKombination:




### Öffentliche Methoden

- AktuelleKombination (EingabeDaten eingabe, boolean horizontal)
- AktuelleKombination (AktuelleKombination kombi2)
- AktuelleKombination ()
- boolean istFrei (String wort, Punkt p, int index, boolean horizontal)
- void add (String wort, Punkt p, int index, boolean horizontal)
- boolean istFertig ()
- int compareTo (AktuelleKombination o)
- Set< Punkt > getAllePunkteZu (char c)
- List< String > getVerfuegbareWoerter ()
- char[][] getKombi ()
- String toString ()
- int hashCode ()
- boolean equals (Object obj)

### 5.1.1 Ausführliche Beschreibung

Diese Klasse definiert eine aktuelle Kombinationsmoeglichkeit von benutzten Worten. Sie enthaelt eine Menge von noch nicht benutzten Worten, und ein char[] der aktuellen Kombination. Ueber die Methoden istFrei und add koennen aus dieser Kombination neue Kombinationen entwickelt werden.

### 5.1.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.1.2.1 raetselErsteller.daten.AktuelleKombination.AktuelleKombination ( EingabeDaten *eingabe,* boolean *horizontal* )

Erstellt eine neue Kombination mit einem Startwort. Als Startwort wird das erste Wort in den Eingabedaten verwendet.

**Parameter**

| | |
|---|---|
| *eingabe* | Die Eingabedatei, welche unter anderen die Worte enthaelt, die in einem Objekt dieser Klassen kombiniert werden sollen. |
| *horizontal* | Gibt an ob das Startwort horizontal oder vertikal eingefuegt werden soll. |

#### 5.1.2.2 raetselErsteller.daten.AktuelleKombination.AktuelleKombination ( **AktuelleKombination** *kombi2* )

Kopierkonstruktor zum Erzeugen einer tiefen Kopie.

#### 5.1.2.3 raetselErsteller.daten.AktuelleKombination.AktuelleKombination ( )

Default Konstruktor.

### 5.1.3 Dokumentation der Elementfunktionen

#### 5.1.3.1 void raetselErsteller.daten.AktuelleKombination.add ( String *wort,* **Punkt** *p,* int *index,* boolean *horizontal* )

Diese Methode fuegt ein neues Wort zur Kombination hinzu. Achtung: vorher sollte ueber istFrei geprueft werden, ob das Wort eingefuegt werden kann.

**Parameter**

| | |
|---|---|
| *wort* | Das Wort, welches in die Kombination eingefuegt werden soll. |
| *p* | Die Position(basierend auf dem Koordinatensystem der Kombination) an welcher der Buchstabe wort[index] eingefuegt werden soll. |
| *index* | An dieser Stelle des Wortes soll das Wort an der Stelle p in die Kombination eingefuegt werden. |
| *horizontal* | Gibt an ob das Wort horizontal oder vertikal eingefuegt werden soll. |

#### 5.1.3.2 int raetselErsteller.daten.AktuelleKombination.compareTo ( **AktuelleKombination** *o* )

Vergleicht die AktuelleKombination mit einem anderen Objekt der selben Klasse. Vergleichskriterium ist die Bewertung. Diese ist definiert als Flaecheninhalt des kombi-Feldes.

**Rückgabe**

1 wenn other<this, -1 wenn other>this, 0 sonst.

#### 5.1.3.3 boolean raetselErsteller.daten.AktuelleKombination.equals ( Object *obj* )

Vergleicht, ob ein objekt dieser Klasse und ein beliebiges anderes Objekt gleich sind. Handelt es sich dabei um ein Objekt der Klasse AktuelleKombination, so gelten sie als gleich, wenn die bereits eingefuegte Kombination und die noch verfuegbaren Worte gleich sind.

#### 5.1.3.4 Set<**Punkt**> raetselErsteller.daten.AktuelleKombination.getAllePunkteZu ( char *c* )

Gibt alle Punkte zurueck, an denen in dieser Kombination ein uebergebenes Zeichen steht.

**Parameter**

| | |
|---|---|
| *c* | Das Zeichen nach dem gesucht werden soll. |

#### 5.1.3.5 char [ ][ ] raetselErsteller.daten.AktuelleKombination.getKombi ( )

Liefert die in der aktuellen Kombination verwendeten Buchstaben als Zeichenarray zurueck.

**Rückgabe**

char[][], welches die Kombination repraesentiert.

#### 5.1.3.6 List<String> raetselErsteller.daten.AktuelleKombination.getVerfuegbareWoerter ( )

Liefert alle Worte zurueck, die noch nicht in dieser Kombination benutzt wurden.

#### 5.1.3.7 int raetselErsteller.daten.AktuelleKombination.hashCode ( )

Erstellt einen Hash-Code zu der aktuellen Kombination.

#### 5.1.3.8 boolean raetselErsteller.daten.AktuelleKombination.istFertig ( )

Gibt zurueck, ob die aktuelle Kombination zu einer vollstaendigen (wenn auch nicht optimalen) Loesung geworden ist.

#### 5.1.3.9 boolean raetselErsteller.daten.AktuelleKombination.istFrei ( String *wort,* Punkt *p,* int *index,* boolean *horizontal* )

Diese Methode prueft ob ein Wort in die aktuelle Kombination eingefuegt werden kann, ohne das es eine Kollision mit einem anderen Buchstaben gibt. Diese Methode prueft nicht, ob die Kombination danach noch gueltig ist(etwa weil das eingefuegte Wort keine Ueberschneidungen mit dem Rest der Kombination aufweist.

**Parameter**

| | |
|---|---|
| *wort* | Das Wort, welches in die Kombination eingefuegt werden soll. |
| *p* | Die Position(basierend auf dem Koordinatensystem der Kombination) an welcher der Buchstabe wort[index] eingefuegt werden soll. |
| *index* | An dieser Stelle des Wortes soll das Wort an der Stelle p in die Kombination eingefuegt werden. |
| *horizontal* | Gibt an ob das Wort horizontal oder vertikal eingefuegt werden soll. |

#### 5.1.3.10 String raetselErsteller.daten.AktuelleKombination.toString ( )

Wandelt die aktuelle Kombination in einen String um. Dargestellt wird zuerst die Anordnung der Buchstaben in Rasterform und dann die Liste der noch verfuegbaren Worte.

**Rückgabe**

Repraesentation der Kombination als String.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/daten/AktuelleKombination.java

## 5.2 raetselErsteller.daten.AusgabeDaten Klassenreferenz

### Öffentliche Methoden

- AusgabeDaten (EingabeDaten eingabeDaten, char[][] c1, char[][] c2)
- String toString ()

### 5.2.1 Ausführliche Beschreibung

Diese Klasse dient zur Repraesentation der Ausgabedaten. Ein Objekt dieser Klasse enthaelt eine optimale Kombination der Worte in der Eingabedatei. Diese Kombination wird in dieser Klasse einmal mit und einmal ohne zusaetzliche Auffuellzeichen gespeichert. Darueber hinaus haelt diese Klasse eine Referenz auf die Eingabedaten aus welchen die Loesung erstellt wurde.

**Autor**

Felix Kibellus

### 5.2.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.2.2.1 raetselErsteller.daten.AusgabeDaten.AusgabeDaten ( EingabeDaten *eingabeDaten,* char *c1[][],* char *c2[][]* )

Erzeugt ein Objekt der Klasse AusgabeDaten.

**Parameter**

| | |
|---|---|
| *eingabeDaten* | die zu den Ausgabedaten gehoerenden Eingabedaten |
| *c1* | die Loesung ohne Auffuellzeichen |
| *c2* | die Loesung mit Auffuellzeichen |

### 5.2.3 Dokumentation der Elementfunktionen

#### 5.2.3.1 String raetselErsteller.daten.AusgabeDaten.toString ( )

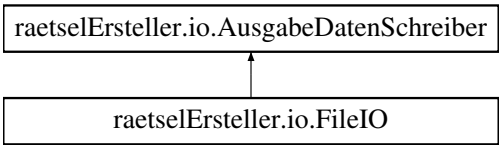
Wandelt die Ausgabedaten in Textform um. Die Darstellung genuegt dem in der Aufgabenanalyse definierten Format, und kann vom Formatierer direkt benutzt werden. Die Ausgabe der Ausgabedaten schiesst die Ausgabe der Eingabedaten mit ein. Deshalb wird in dieser Methode an die toString-Methode der Klasse EingabeDaten deligiert.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/daten/AusgabeDaten.java

## 5.3 raetselErsteller.io.AusgabeDatenSchreiber Schnittstellenreferenz

Klassendiagramm für raetselErsteller.io.AusgabeDatenSchreiber:



### Öffentliche Methoden

- void schreibeAusgabe (AusgabeDaten out) throws FileIOException

### 5.3.1 Ausführliche Beschreibung

Dieses Interface definiert die Schnittstelle zum Schreiben der Ausgabedaten. Durch die Abstraktion dieser Schnittstelle muss der Controller nicht wesentlich veraendert werden, um die Ausgabefunktionalitaet zu aendern.

**Autor**

Felix Kibellus

### 5.3.2 Dokumentation der Elementfunktionen

#### 5.3.2.1 void raetselErsteller.io.AusgabeDatenSchreiber.schreibeAusgabe ( AusgabeDaten *out* ) throws FileIOException

Gibt die Ausgabedaten aus. Wie die Ausgabe genau funktioniert (Datei, Konsole, Netzwerk,...) muss durch eine konkrete Klasse entschieden werden, welche dieses Interface implementiert.

**Parameter**

| | |
|---|---|
| *outputData* | Die Ausgabedaten, welche die fertigen Raetsel beinhalten. |

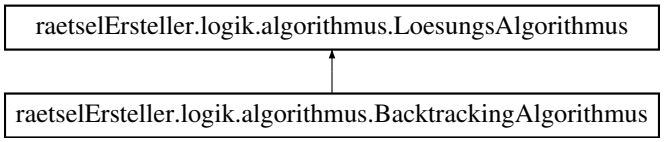
Implementiert in raetselErsteller.io.FileIO.

Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/AusgabeDatenSchreiber.java

## 5.4 raetselErsteller.logik.algorithmus.BacktrackingAlgorithmus Klassenreferenz

Klassendiagramm für raetselErsteller.logik.algorithmus.BacktrackingAlgorithmus:




### Öffentliche Methoden

- BacktrackingAlgorithmus (boolean tiefensuche)
- AusgabeDaten loese (EingabeDaten eingabe) throws KeineLoesungException

### Statische öffentliche Attribute

- static Logger **logger**

### 5.4.1 Ausführliche Beschreibung

Diese Klasse stellt einen Loesungsalgorithmus zum Erstellen von wortraetseln dar. Sie ist in der Lage, zu einer gegebenen Menge von Worten ein Raetsel mit minimaler Ausdehnung zu finden. Dazu wird ein Backtracking--Algorithmus verwendet, der saemtliche Kombinations- moeglichkeiten ausprobiert, und somit einen Suchbaum aufbaut. Da in diesem alle moeglichen Loesungen enthalten sind wird die beste Loesung gefunden. Es besteht die

Moeglichkeit diesen Suchbaum in Breiten- oder Tiefensuche zu durchlaufen. Empfohlen ist die Verwendung einer Tiefensuche.

**Autor**

Felix Kibellus

### 5.4.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.4.2.1 raetselErsteller.logik.algorithmus.BacktrackingAlgorithmus.BacktrackingAlgorithmus ( boolean *tiefensuche* )

Erstellt ein Objekt vom Typ BacktrackingAlgorithmus

**Parameter**

| | |
|---|---|
| *tiefensuche* | gibt an ob der Algorithmus Tiefensuche oder Breitensuche verwenden soll |

### 5.4.3 Dokumentation der Elementfunktionen

#### 5.4.3.1 AusgabeDaten raetselErsteller.logik.algorithmus.BacktrackingAlgorithmus.loese ( EingabeDaten *eingabe* ) throws KeineLoesungException

Diese Methode liefert ausgehend von eingelesenen Eingabedaten ein Raetsel mit minimaler Ausdehnung. Dazu wird ein Backtracking-Ansatz werdende.

**Parameter**

| | |
|---|---|
| *eingabe* | Ein Objekt der Klasse EingabeDaten, welches die Worte enthaelt, welche zu einem Raetsel kombiniert werden sollen. |

**Rückgabe**

Das fertige Raetsel mit minimaler Ausdehnung.

Implementiert raetselErsteller.logik.algorithmus.LoesungsAlgorithmus.

### 5.4.4 Dokumentation der Datenelemente

#### 5.4.4.1 Logger raetselErsteller.logik.algorithmus.BacktrackingAlgorithmus.logger [static]

**Initialisierung:**

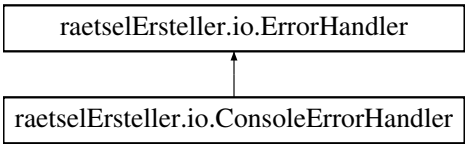
```
= Logger
            .getLogger(BacktrackingAlgorithmus.class.getName())
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/logik/algorithmus/BacktrackingAlgorithmus.java

## 5.5 raetselErsteller.io.ConsoleErrorHandler Klassenreferenz

Klassendiagramm für raetselErsteller.io.ConsoleErrorHandler:




### Öffentliche Methoden

- void zeigeFehler (String msg)

### 5.5.1 Ausführliche Beschreibung

Diese Klasse kann zur Fehlerausgabe benutzt werden. Sie schreibt auftretende Fehler auf StandardErr, also auf die Fehleraushabe.

**Autor**

Felix Kibellus

### 5.5.2 Dokumentation der Elementfunktionen

#### 5.5.2.1 void raetselErsteller.io.ConsoleErrorHandler.zeigeFehler ( String *msg* )

Schreibt einen Fehler auf die Fehleraushabe.

**Parameter**

| | |
|---|---|
| *msg* | Die Fehlernachricht, welche ausgegben werden soll. |

Implementiert raetselErsteller.io.ErrorHandler.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/ConsoleErrorHandler.java

## 5.6 raetselErsteller.logik.Controller Klassenreferenz

### Öffentliche Methoden

- Controller (String inPath, String outPath)
- Controller (String inPath)
- void loese ()

### Statische öffentliche Attribute

- static Logger **logger**

### Statische, geschützte Attribute

- static boolean **timestamp**
- static boolean **tiefensuche** = true

### 5.6.1 Ausführliche Beschreibung

Diese Klasse stellt den zentralen Punkt der Programmarchitektur dar. Sie verwaltet die einzelnen Aufgaben und leitet Funktionsaufrufe an die entsprechenden Schnittstellen von Daten-, Ein/Ausgabeschicht und Algorithmus weiter.

**Autor**

Felix Kibellus

### 5.6.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.6.2.1 raetselErsteller.logik.Controller.Controller ( String *inPath,* String *outPath* )

Initialisiert die Controller-Klasse. In diesem Konstruktor wird festgelegt, wie die Ein- und Ausgabe erfolgen soll. Sollte keine Interaktion ueber Files mehr erwuenscht sein, so muss hier die Klasse FileIO durch eine eigene Klasse, welche die noetigen Interfaces implementiert, ersetzt werden. Ausserdem wird in diesem Konstruktor der ErrorHandler gesetzt. Sollte keine Fehlerausgabe auf StandardError mehr erwuenscht sein, so muss das betreffende Objekt durch einen alternativen ErrorHandler angepasst werden.

**Parameter**

| | |
|---|---|
| *inPath* | Der Pfad zur Eingabedatei |
| *outPath* | Der Pfad zur Ausgabedatei |

#### 5.6.2.2 raetselErsteller.logik.Controller.Controller ( String *inPath* )

Wenn dieser Konstruktor aufgerufen wird, wird der Ausgabepfad ueber die Funktion getDefaultOutputPath bestimmt.

### 5.6.3 Dokumentation der Elementfunktionen

#### 5.6.3.1 void raetselErsteller.logik.Controller.loese ( )

In dieser Methode werden zunaechst die Eingabedaten aus der Eingabedatei gelesen. Danach wird der Loesungsalgorithmus mit den Eingabedaten ausgefuehrt. Zum Schluss werden die Ausgabedaten in eine Ausgabedatei geschrieben. Wenn es bei einem der beschriebenen Schritten zu einem Fehler kommt, wird die passende Fehlermeldung ueber den errorHandler ausgegeben.

### 5.6.4 Dokumentation der Datenelemente

#### 5.6.4.1 Logger raetselErsteller.logik.Controller.logger [static]

**Initialisierung:**

```
= Logger.getLogger(Controller.class
            .getName())
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/logik/Controller.java

## 5.7 raetselErsteller.daten.EingabeDaten Klassenreferenz

### Öffentliche Methoden

- EingabeDaten (List< String > kommentar, List< String > woerter)

- List< String > getKommentare ()
- List< String > getWorte ()
- String toString ()

### 5.7.1 Ausführliche Beschreibung

Diese Klasse dient der Repraesentation der Eingabedaten. Ein Objekt dieser Klasse enthaelt: -Liste der Kommentare ausn der Eingabedatei -List der Worte, welche in einer Kombination verwendet werden sollen

**Autor**

Felix Kibellus

### 5.7.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.7.2.1 raetselErsteller.daten.EingabeDaten.EingabeDaten ( List< String > *kommentar*, List< String > *woerter* )

Erstellt ein Objekt der Klasse EingabeDaten

**Parameter**

| | |
|---|---|
| *kommentar* | Liste der Kommentare aus der Eingabedatei |
| *woerter* | Liste der Worte aus der Eingabedatei |

### 5.7.3 Dokumentation der Elementfunktionen

#### 5.7.3.1 List<String> raetselErsteller.daten.EingabeDaten.getKommentare ( )

Liefert die Liste der Kommentare zurueck. Diese Methode wird in der aktuellen implementierung nicht verwendet, wird aber benoetigt, wenn ein alternativer Formatierer die Eingabedaten formatieren soll.

#### 5.7.3.2 List<String> raetselErsteller.daten.EingabeDaten.getWorte ( )

Liefert die Liste der Worte zurueck. Diese Methode wird in der aktuellen implementierung nicht verwendet, wird aber benoetigt, wenn ein alternativer Formatierer die Eingabedaten formatieren soll.

#### 5.7.3.3 String raetselErsteller.daten.EingabeDaten.toString ( )

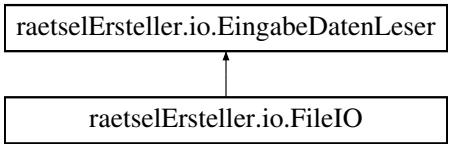
Wandelt die Eingabedaten in einen String um. Die Darstellung genuegt dem in der Aufgabenanalyse beschriebenen Format.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/daten/EingabeDaten.java

## 5.8 raetselErsteller.io.EingabeDatenLeser Schnittstellenreferenz

Klassendiagramm für raetselErsteller.io.EingabeDatenLeser:



### Öffentliche Methoden

- EingabeDaten leseEingabe () throws FileIOException, FormatException

### 5.8.1 Ausführliche Beschreibung

Dieses Interface definiert die Schnittstelle zum Lesen der Eingabedaten. Durch die Abstraktion dieser Schnittstelle muss der Controller nicht wesentlich veraendert werden, um die Eingabefunktionalitaet zu aendern.

**Autor**

Felix Kibellus

### 5.8.2 Dokumentation der Elementfunktionen

#### 5.8.2.1 EingabeDaten raetselErsteller.io.EingabeDatenLeser.leseEingabe ( ) throws FileIOException, FormatException

Liesst die Eingabedaten ein. Wie die Eingabe genau funktioniert (Datei, Konsole, Netzwerk,...) muss durch eine konkrete Klasse entschieden werden, welche dieses Interface implementiert.

**Rückgabe**

Die Eingabedaten, welche die Worte enthalten, die zu einem fertigen Raetsel kombiniert werden sollen

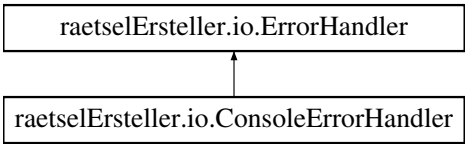
Implementiert in raetselErsteller.io.FileIO.

Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/EingabeDatenLeser.java

## 5.9 raetselErsteller.io.ErrorHandler Schnittstellenreferenz

Klassendiagramm für raetselErsteller.io.ErrorHandler:




### Öffentliche Methoden

- void zeigeFehler (String fehler)

### 5.9.1 Ausführliche Beschreibung

Dieses Interface definiert die Schnittstelle zum Ausgeben von Fehlern. Wenn ein Fehler auftritt, leitet der Controller die zugehoerige Fehlernachricht an diese Schnittstelle weiter. Eine Klasse, welche dieses Interface implementiert muss die Nachricht dann geeignet verarbeiten. (StandardErr, Datei, ...)

**Autor**

Felix Kibellus

### 5.9.2 Dokumentation der Elementfunktionen

#### 5.9.2.1 void raetselErsteller.io.ErrorHandler.zeigeFehler ( String *fehler* )

Diese Methode stellt die Schnittstelle zur Fehlerausgabe dar.

**Parameter**

| | |
|---|---|
| *msg* | Eine Nachricht, welche den aufgetretenden Fehler beschreibt. |

Implementiert in raetselErsteller.io.ConsoleErrorHandler.

Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/ErrorHandler.java

## 5.10 raetselErsteller.io.FileIOException.ErrorType Enum-Referenz

### Öffentliche Attribute

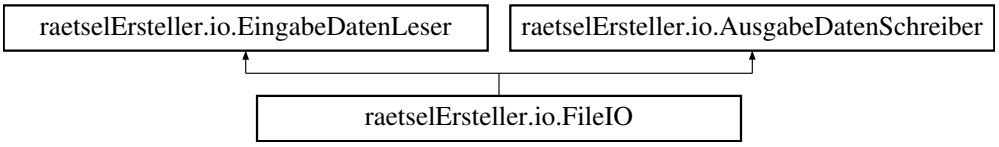
- **InputFileNotFound**
- **OutputPathInvalid**

Die Dokumentation für diesen enum wurde aus der folgenden Datei generiert:

- GroProCode/src/raetselErsteller/io/FileIOException.java

## 5.11 raetselErsteller.io.FileIO Klassenreferenz

Klassendiagramm für raetselErsteller.io.FileIO:




### Öffentliche Methoden

- FileIO (String inPfad, String outPfad, Formatierer formatierer)
- void schreibeAusgabe (AusgabeDaten out) throws FileIOException
- EingabeDaten leseEingabe () throws FileIOException, FormatException

### Statische öffentliche Attribute

- static Logger **logger**

### 5.11.1 Ausführliche Beschreibung

Diese Klasse kann zum Lesen und Schreiben von Dateien verwendet werden. Sie liest die Eingabedaten aus einer Eingabedatei und schreibt die Ausgabedaten in eine Ausgabedatei. Das Format von Ein- und Ausgabedatei kann ueber einen Formatierer angepasst werden.

**Autor**

Felix Kibellus

### 5.11.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.11.2.1 raetselErsteller.io.FileIO.FileIO ( String *inPfad,* String *outPfad,* Formatierer *formatierer* )

Erstellt ein Objekt der Klasse FileIO.

**Parameter**

| | |
|---|---|
| *inPfad* | der Pfad zur Eingabedatei |
| *outPfad* | der Pfad zur Ausgabedatei |
| *formatierer* | ein Formatierer, der regelt wie Ein- und Ausgabe formatiert werden soll |

### 5.11.3 Dokumentation der Elementfunktionen

#### 5.11.3.1 EingabeDaten raetselErsteller.io.FileIO.leseEingabe ( ) throws FileIOException, FormatException

Liest die Eingabedaten aus einer Datei und gibt diese zurueck. Dazu werden zuerst die Zeilen aus der Datei eingelesen, und dann vom Formatierer in eine Eingabedatei umgewandelt.

**Rückgabe**

Aus der Datei gelesene Eingabedaten

Implementiert raetselErsteller.io.EingabeDatenLeser.

#### 5.11.3.2 void raetselErsteller.io.FileIO.schreibeAusgabe ( AusgabeDaten *out* ) throws FileIOException

Schreibt die uebergebenen Ausgabedaten in eine Datei

**Parameter**

| | |
|---|---|
| *out* | Ausgabedaten, welche in eine Datei geschrieben werden sollen |

Implementiert raetselErsteller.io.AusgabeDatenSchreiber.

### 5.11.4 Dokumentation der Datenelemente

#### 5.11.4.1 Logger raetselErsteller.io.FileIO.logger [static]

**Initialisierung:**

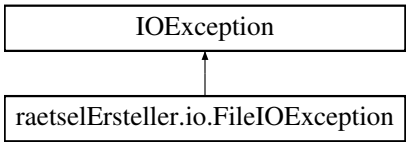
```
= Logger.getLogger(FileIO.class
            .getName())
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/FileIO.java

## 5.12 raetselErsteller.io.FileIOException Klassenreferenz

Klassendiagramm für raetselErsteller.io.FileIOException:




### Klassen

- enum ErrorType

### Öffentliche Methoden

- FileIOException (ErrorType type)
- ErrorType getType ()

### 5.12.1 Ausführliche Beschreibung

Diese Klasse wird verwendet um Aussnahmen beim Lesen und Schreiben von Dateien zu behandeln. Sie ist in der Lage, die beiden Faelle Eingabedatei existiert nicht und Ausgabepfad ist ungueltig zu repraesentieren. Ueber die Methode getType() kann abgefragt werden, welcher von beiden Faellen eingetreten ist.

**Autor**

Felix Kibellus

### 5.12.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.12.2.1 raetselErsteller.io.FileIOException.FileIOException ( ErrorType *type* )

Erzeugt ein Objekt vom Typ FileIOException

**Parameter**

| | |
|---|---|
| *type* | Hier muss angegeben werden, ob die Eingabedatei nicht existiert oder ob der Pfad zur Ausgabedatei ungueltig ist |

### 5.12.3 Dokumentation der Elementfunktionen

#### 5.12.3.1 ErrorType raetselErsteller.io.FileIOException.getType ( )

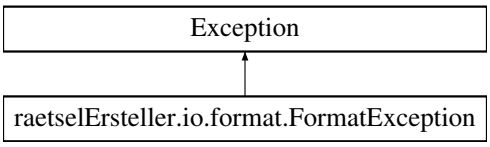
Liefert zurueck, welcher Fehlerfall bei der Ein- oder Ausgabe aufgetreten ist.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/FileIOException.java

## 5.13 raetselErsteller.io.format.FormatException Klassenreferenz

Klassendiagramm für raetselErsteller.io.format.FormatException:




### Klassen

- enum GrundType

### Öffentliche Methoden

- FormatException (int line, GrundType type)
- int getLine ()
- GrundType getType ()

## 5.13.1 Ausführliche Beschreibung

Diese Klasse wird zur Repraesentation von Aussnahmen verwendet, welche beim Parsen der Eingabedaten entstehen koennen. Ueber getLine() kann abgefragt werden, in welcher Zeile der Fehler aufgetreten ist. Ueber getType() kann abgefragt werden, welcher Fehler genau aufgetreten ist.

**Autor**

Felix Kibellus

## 5.13.2 Beschreibung der Konstruktoren und Destruktoren

### 5.13.2.1 raetselErsteller.io.format.FormatException.FormatException ( int *line,* GrundType *type* )

Erzeugt ein Objekt der Klasse FormatException

**Parameter**

| | |
|---|---|
| *line* | Zeile in der der Fehler aufgetreten ist |
| *type* | genauer Grund fuer den Fehlers. |

## 5.13.3 Dokumentation der Elementfunktionen

### 5.13.3.1 int raetselErsteller.io.format.FormatException.getLine ( )

Gibt die Zeile an, in der der Fehler aufgetreten ist.

### 5.13.3.2 GrundType raetselErsteller.io.format.FormatException.getType ( )

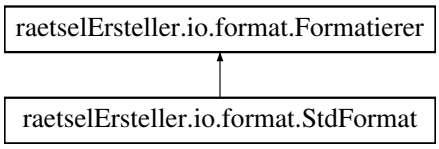
Gibt den genauen Fehlergrund an.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/format/FormatException.java

## 5.14 raetselErsteller.io.format.Formatierer Schnittstellenreferenz

Klassendiagramm für raetselErsteller.io.format.Formatierer:




### Öffentliche Methoden

- String formatiere (AusgabeDaten out)
- EingabeDaten parse (List< String > in) throws FormatException

### 5.14.1 Ausführliche Beschreibung

Diese Schnittstelle dient der Abstraktion des Formates von Ein- und Ausgabedatei.

**Autor**

Felix Kibellus

### 5.14.2 Dokumentation der Elementfunktionen

#### 5.14.2.1 String raetselErsteller.io.format.Formatierer.formatiere ( AusgabeDaten *out* )

Formatiert die Ausgabedaten zu einem Text, der in die Ausgabedatei geschrieben werden kann.

Implementiert in raetselErsteller.io.format.StdFormat.

#### 5.14.2.2 EingabeDaten raetselErsteller.io.format.Formatierer.parse ( List< String > *in* ) throws FormatException

Erzeugt aus den aus der Eingabedatei eingelesenen Zeilen ein Objekt vom Typ Eingabedaten.

Implementiert in raetselErsteller.io.format.StdFormat.

Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/format/Formatierer.java

## 5.15 raetselErsteller.io.format.FormatException.GrundType Enum-Referenz

### Öffentliche Attribute

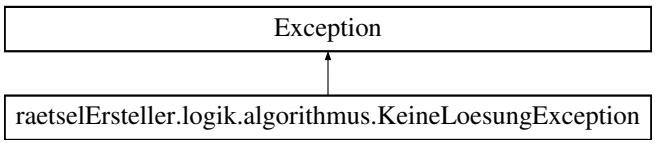
- **KeinBuchstabe**
- **NichtGenugWorte**
- **Leerzeile**

Die Dokumentation für diesen enum wurde aus der folgenden Datei generiert:

- GroProCode/src/raetselErsteller/io/format/FormatException.java

## 5.16 raetselErsteller.logik.algorithmus.KeineLoesungException Klassenreferenz

Klassendiagramm für raetselErsteller.logik.algorithmus.KeineLoesungException:




### 5.16.1 Ausführliche Beschreibung

Diese Exception repraesentiert den Ausnahmefall, dass aus den angegebenen Worten kein Raetsel gebildet werden kann. Dies ist beispielsweise der Fall, wenn die Worte disjunkte Buchstabenmengen enthalten.

**Autor**

Felix Kibellus

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/logik/algorithmus/KeineLoesungException.java

## 5.17 raetselErsteller.Konstanten Klassenreferenz

### Öffentliche, statische Methoden

- static String **buchstabeInZeile** (int line)

### Statische öffentliche Attribute

- static final boolean **useMap** = false
- static final String **dateiendungIn** = ".in"
- static final String **dateiendungOut** = ".out"
- static final String **inputData** = "EINGABEDATEI"
- static final String **projectname**
- static final String **beschreibung**
- static final String **log**
- static final String **output**
- static final String **input**
- static final String **timeStamp**
- static final String **algo**
- static String **keinSpeicher**
- static String **leerzeile**
- static final String **inputFileNotFound**
- static final String **nichtGenugWorte**
- static final String **outputPathInvalid**
- static final String **unknownIoErr**
- static final String **unknownFormatErr**
- static final String **keineLoesung**
- static final String **laufzeit** = "Laufzeit: "
- static final String **raetselVersteckt** = "Raetsel versteckt"
- static final String **raetselNichtVersteckt**

### 5.17.1 Ausführliche Beschreibung

In dieser Klasse sind Konstanten definiert. Diese lassen sich in 4 Kategorien unterteilen: -Programmeinstellungen -Hiletext fuer den Argparser -Fehlermeldungen -Ausgabe Die Programmeinstellungen koennen geaendert werden, um das Verhalten des Programms zu beeinflussen. Sollte das Programm in einer anderen Sprache benuetigt werden, genuegt es die Texte in dieser Datei auszutauschen.

**Autor**

Felix Kibellus

### 5.17.2 Dokumentation der Datenelemente

#### 5.17.2.1 final String raetselErsteller.Konstanten.algo [static]

**Initialisierung:**

```
= "Legt fest welcher Algoritmus "
            + "verwendet werden soll. Zur Auswahl stehen Tiefensuche "
            + "und Breitensuche. Es wird empfohlen Tiefensuche zu "
            + "benutzen."
```

#### 5.17.2.2 final String raetselErsteller.Konstanten.beschreibung [static]

**Initialisierung:**

```
= "Dieses Programm kann "
            + "zum Erstellen von Raetseln genutzt werden."
```

#### 5.17.2.3 final String raetselErsteller.Konstanten.input [static]

**Initialisierung:**

```
= "Hier muss der Pfad zur "
            + "Eingabedatei angegeben werden. Hier gibt es keinen "
            + "Default-Wert. Die Eingabedatei muss angegeben werden."
```

#### 5.17.2.4 final String raetselErsteller.Konstanten.inputFileNotFound [static]

**Initialisierung:**

```
= "Die Eingabedatei "
            + "konnte nicht gefunden werden. Das Programm wird "
            + "abgebrochen."
```

#### 5.17.2.5 final String raetselErsteller.Konstanten.keineLoesung [static]

**Initialisierung:**

```
= "Zu den eingegebenen "
            + "Worten " + "kann kein Raetsel gebildet werden."
```

#### 5.17.2.6 String raetselErsteller.Konstanten.keinSpeicher [static]

**Initialisierung:**

```
= "Es steht nicht genug Speicher"
            + " zur Verfuegung. Das Programm wird abgebrochen."
```

#### 5.17.2.7 String raetselErsteller.Konstanten.leerzeile [static]

**Initialisierung:**

```
= "In der Eingabedatei befindet" +
            " sich eine Leerzeile."
```

#### 5.17.2.8 final String raetselErsteller.Konstanten.log [static]

**Initialisierung:**

```
= "Über diesen Parameter kann das "
            + "Logging aktiviert werden. Moegliche Parameter sind ALL, "
            + "CONFIG, FINE, FINER, FINEST, INFO, OFF, SEVERE und "
            + "WARNING. Default ist OFF. FINER zeigt alle "
            + "Methodenaufrufe an, mit FINEST kann die Funktionalität "
            + "des Algorithmus nachvollzogen werde,"
```

#### 5.17.2.9 final String raetselErsteller.Konstanten.nichtGenugWorte [static]

**Initialisierung:**

```
= "Die Eingabedatei "
            + "muss mindestens 2 Woerter enthalten, damit das Problem "
            + "sinnvoll zu bearbeiten ist."
```

#### 5.17.2.10 final String raetselErsteller.Konstanten.output [static]

**Initialisierung:**

```
= "Dieser Parameter legt den "
            + "Pfad fuer die Ausgabedatei fest. Default ist Pfad und "
            + "Name der Eingabedatei. Endet die Eingabedatei auf .in"
            + " wird .in durch .out ersetzt. Andernfalls wird nur .out "
            + "an den Namen der " + "Eingabedatei angehaengt."
```

#### 5.17.2.11 final String raetselErsteller.Konstanten.outputPathInvalid [static]

**Initialisierung:**

```
= "Es liegt ein "
            + "Problem mit " + "dem Pfad der Ausgabedatei vor."
```

#### 5.17.2.12 final String raetselErsteller.Konstanten.projectname [static]

**Initialisierung:**

```
= "java -jar"
            + " raetselErsteller.jar"
```

#### 5.17.2.13 final String raetselErsteller.Konstanten.raetselNichtVersteckt [static]

**Initialisierung:**

```
= "Raetsel "
            + "nicht versteckt"
```

#### 5.17.2.14 final String raetselErsteller.Konstanten.timeStamp [static]

**Initialisierung:**

```
= "Gibt die benoetigte "
            + "Laufzeit des Algorithmus aus. Die zeit zum Einlesen und "
            + "Ausgeben ist nicht " + "mit eingeschlossen"
```

#### 5.17.2.15 final String raetselErsteller.Konstanten.unknownFormatErr [static]

**Initialisierung:**

```
= "Unbekannter Fehler"
            + " formatieren der Ein- oder Ausgabe."
```

#### 5.17.2.16 final String raetselErsteller.Konstanten.unknownIoErr [static]

**Initialisierung:**

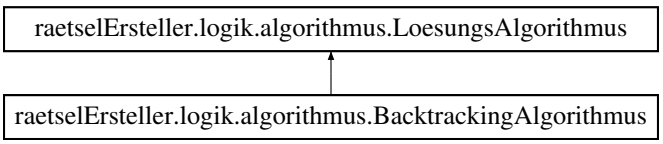
```
= "Unbekannter Fehler beim "
            + "Lesen oder Schreiben von Dateien."
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/Konstanten.java

## 5.18 raetselErsteller.logik.algorithmus.LoesungsAlgorithmus Schnittstellenreferenz

Klassendiagramm für raetselErsteller.logik.algorithmus.LoesungsAlgorithmus:




### Öffentliche Methoden

- AusgabeDaten loese (EingabeDaten in) throws KeineLoesungException

### 5.18.1 Ausführliche Beschreibung

Dieses Interface dient der abstraktion des Loesealgorithmus. Es definiert die Schnittstelle fuer einen Algorithmus, welcher aus den eingelesenen Eingabedaten die benoetigten Ausgabedaten generiert.

**Autor**

Felix Kibellus

### 5.18.2 Dokumentation der Elementfunktionen

#### 5.18.2.1 AusgabeDaten raetselErsteller.logik.algorithmus.LoesungsAlgorithmus.loese ( EingabeDaten *in* ) throws KeineLoesungException

Kombiniert die Worte aus den Eingabedaten zu einem Raetsel mit minimaler Ausdehnung(Flaecheninhalt).

**Parameter**

| | |
|---|---|
| *inputData* | Ein Objekt der Klasse EingabeDaten, welches die Worte enthaelt, welche zu einem Raetsel kombiniert werden sollen. |

**Rückgabe**

Das fertige Raetsel mit minimaler Ausdehnung.

Implementiert in raetselErsteller.logik.algorithmus.BacktrackingAlgorithmus.

Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/logik/algorithmus/LoesungsAlgorithmus.java

## 5.19 raetselErsteller.logik.Main Klassenreferenz

### Öffentliche, statische Methoden

- static void main (String[] args)

### 5.19.1 Ausführliche Beschreibung

Diese Klasse beinhaltet den Startpunkt des Programms und die Logik zum Parsen von Argumenten. Ausserdem wird in dieser Klasse das Logging konfiguriert und die benoetigten Logger auf das vom Benutzer gewuenscht Level eingestellt.

**Autor**

Felix Kibellus

### 5.19.2 Dokumentation der Elementfunktionen

#### 5.19.2.1 static void raetselErsteller.logik.Main.main ( String[] *args* ) [static]

Startpunkt des Programms. Zunaechst werden die uebergebenen Argumente analysiert und auf Korrektheit geprueft. Dann wird das Programm, wie in den Argumenten beschrieben, konfiguriert. Anschliessend wird es gestartet.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/logik/Main.java

## 5.20 raetselErsteller.daten.Punkt Klassenreferenz

### Öffentliche Methoden

- Punkt (int x, int y)
- Punkt (Punkt p)
- String toString ()
- int **hashCode** ()
- boolean **equals** (Object obj)

### Öffentliche Attribute

- int **x**
- int **y**

### 5.20.1 Ausführliche Beschreibung

Diese Klasse repraesentiert einen Punkt (x,y) im R^2 und wird verwendet um Positionen von Buchstaben in einer Kombination zu beschreiben.

**Autor**

Felix Kibellus

### 5.20.2 Beschreibung der Konstruktoren und Destruktoren

#### 5.20.2.1 raetselErsteller.daten.Punkt.Punkt ( int *x*, int *y* )

Erstellt ein Objekt der Klasse Punkt

**Parameter**

| | |
|---|---|
| *x* | Die x-Koordinate des Punktes |
| *y* | Die y-Koordinate des Punktes |

#### 5.20.2.2 raetselErsteller.daten.Punkt.Punkt ( Punkt *p* )

Erstellt eine tiefe Kopie eines Punktes

### 5.20.3 Dokumentation der Elementfunktionen

#### 5.20.3.1 String raetselErsteller.daten.Punkt.toString ( )

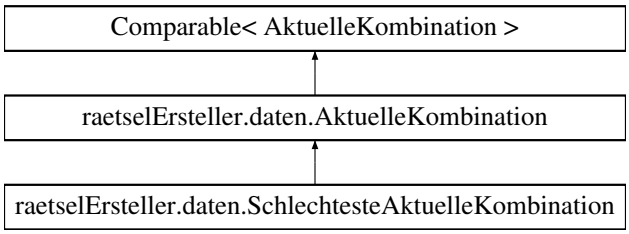
Wandelt des Punkt in einen String um. Format: (x,y)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/daten/Punkt.java

## 5.21 raetselErsteller.daten.SchlechtesteAktuelleKombination Klassenreferenz

Klassendiagramm für raetselErsteller.daten.SchlechtesteAktuelleKombination:




### Öffentliche Methoden

- int **compareTo** (AktuelleKombination o)

### 5.21.1 Ausführliche Beschreibung

DIes ist eine aktuelle Kombination, welche immer schlechter als eine andere aktuelle Kombination ist. Sie wird verwendet um die aktuell beste Kombination im Loesungsalgorithmus zu initialisieren.

**Autor**

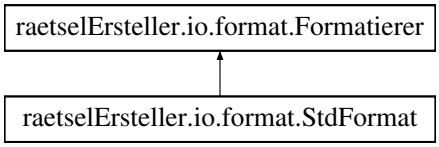
Felix Kibellus

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/daten/SchlechtesteAktuelleKombination.java

## 5.22 raetselErsteller.io.format.StdFormat Klassenreferenz

Klassendiagramm für raetselErsteller.io.format.StdFormat:




### Öffentliche Methoden

- String formatiere (AusgabeDaten data)
- EingabeDaten parse (List< String > inputString) throws FormatException

### Statische öffentliche Attribute

- static Logger **logger**

### 5.22.1 Ausführliche Beschreibung

Diese Klasse kann zum formatieren von Ausgabedaten und zum parsen von Eingabedaten verwendet werden. Das verwendete Format genuegt den in der Aufgabenanalyse definierten Kriterien.

**Autor**

Felix Kibellus

### 5.22.2 Dokumentation der Elementfunktionen

#### 5.22.2.1 String raetselErsteller.io.format.StdFormat.formatiere ( AusgabeDaten *data* )

Formatiert die Ausgabedaten. Dazu wird das in der toString-Methode definierte format gewaehlt.

**Parameter**

| | |
|---|---|
| *data* | Die Ausgabedaten, welche formatiert werden sollen. |

Implementiert raetselErsteller.io.format.Formatierer.

#### 5.22.2.2 EingabeDaten raetselErsteller.io.format.StdFormat.parse ( List< String > *inputString* ) throws FormatException

Diese Methode erzeugt aus den aus der Eingabedatei gelesenen Zeilen ein Objekt der Klasse EingabeDaten. Dabei werden die Kommentare gespeichert, sodass diese spaeter ausgegeben werden koennen. Ausserdem werden Fehler in den Eingabedaten erkannt. Beispielsweise koennen weniger als 2 Worte angegeben worden sein, oder ein Wort nicht ausschliesslich aus Buchstaben bestehen.

Implementiert raetselErsteller.io.format.Formatierer.

### 5.22.3 Dokumentation der Datenelemente

#### 5.22.3.1 Logger raetselErsteller.io.format.StdFormat.logger [static]

**Initialisierung:**

```
= Logger.getLogger(StdFormat.class
            .getName())
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- GroProCode/src/raetselErsteller/io/format/StdFormat.java

# Index