

DistanzSpiel

Erzeugt von Doxygen 1.8.9.1

Son Jun 21 2015 22:49:06

Inhaltsverzeichnis

1	Hierarchie-Verzeichnis	1
1.1	Klassenhierarchie	1
2	Klassen-Verzeichnis	3
2.1	Auflistung der Klassen	3
3	Datei-Verzeichnis	5
3.1	Auflistung der Dateien	5
4	Klassen-Dokumentation	7
4.1	Feld Klassenreferenz	7
4.1.1	Ausführliche Beschreibung	9
4.1.2	Beschreibung der Konstruktoren und Destruktoren	9
4.1.2.1	Feld	9
4.1.2.2	Feld	9
4.1.2.3	Feld	9
4.1.3	Dokumentation der Elementfunktionen	9
4.1.3.1	delStein	9
4.1.3.2	getBesetzt	9
4.1.3.3	getGast	9
4.1.3.4	getPos	10
4.1.3.5	setStein	10
4.1.4	Dokumentation der Datenelemente	10
4.1.4.1	besetzt	10
4.1.4.2	gast	10
4.1.4.3	pos	10
4.2	GUI Klassenreferenz	10
4.2.1	Ausführliche Beschreibung	12
4.2.2	Beschreibung der Konstruktoren und Destruktoren	12
4.2.2.1	GUI	12
4.2.3	Dokumentation der Elementfunktionen	12
4.2.3.1	Spieler	12

4.2.3.2	zeichneAnleitung	12
4.2.3.3	zeichneSpielfeld	12
4.2.3.4	zeichneZug	12
4.2.4	Dokumentation der Datenelemente	12
4.2.4.1	brett	12
4.2.4.2	Klsw	13
4.3	KI Klassenreferenz	13
4.3.1	Ausführliche Beschreibung	15
4.3.2	Beschreibung der Konstruktoren und Destruktoren	15
4.3.2.1	KI	15
4.3.2.2	~KI	15
4.3.3	Dokumentation der Elementfunktionen	15
4.3.3.1	getBrett	15
4.3.3.2	getTeam	15
4.3.3.3	mergeStrategie	16
4.3.3.4	mergeStrategie	17
4.3.3.5	nexZug	18
4.3.3.6	seachBestZug	18
4.3.4	Dokumentation der Datenelemente	19
4.3.4.1	abrett	19
4.3.4.2	anzstrat	19
4.3.4.3	dv	19
4.3.4.4	nZug	19
4.3.4.5	strat	19
4.3.4.6	t	19
4.4	Koenig Klassenreferenz	20
4.4.1	Beschreibung der Konstruktoren und Destruktoren	22
4.4.1.1	Koenig	22
4.4.1.2	Koenig	22
4.4.2	Dokumentation der Elementfunktionen	22
4.4.2.1	setGeffangen	22
4.4.2.2	ziehenach	22
4.5	Possition Strukturreferenz	23
4.5.1	Ausführliche Beschreibung	24
4.5.2	Beschreibung der Konstruktoren und Destruktoren	24
4.5.2.1	Possition	24
4.5.2.2	Possition	24
4.5.3	Dokumentation der Elementfunktionen	24
4.5.3.1	operator==	24
4.5.4	Dokumentation der Datenelemente	24

4.5.4.1	x	24
4.5.4.2	y	24
4.6	SfH Klassenreferenz	24
4.6.1	Ausführliche Beschreibung	27
4.6.2	Beschreibung der Konstruktoren und Destruktoren	27
4.6.2.1	SfH	27
4.6.2.2	~SfH	27
4.6.3	Dokumentation der Elementfunktionen	27
4.6.3.1	bewerten	27
4.7	SfK Klassenreferenz	28
4.7.1	Ausführliche Beschreibung	31
4.7.2	Beschreibung der Konstruktoren und Destruktoren	31
4.7.2.1	SfK	31
4.7.2.2	SfK	31
4.7.2.3	~SfK	31
4.7.3	Dokumentation der Elementfunktionen	31
4.7.3.1	bewerten	31
4.8	SpielBrett Klassenreferenz	33
4.8.1	Beschreibung der Konstruktoren und Destruktoren	34
4.8.1.1	SpielBrett	34
4.8.1.2	SpielBrett	34
4.8.1.3	~SpielBrett	35
4.8.2	Dokumentation der Elementfunktionen	35
4.8.2.1	getFeld	35
4.8.2.2	getSchwarz	35
4.8.2.3	getWeis	36
4.8.2.4	initBrett	36
4.8.3	Dokumentation der Datenelemente	36
4.8.3.1	Brett	36
4.8.3.2	dimENSION	36
4.8.3.3	schwarz	36
4.8.3.4	weis	36
4.9	SrH Klassenreferenz	36
4.9.1	Ausführliche Beschreibung	39
4.9.2	Beschreibung der Konstruktoren und Destruktoren	39
4.9.2.1	SrH	39
4.9.2.2	~SrH	39
4.9.3	Dokumentation der Elementfunktionen	39
4.9.3.1	bewerten	39
4.10	SsK Klassenreferenz	40

4.10.1 Ausführliche Beschreibung	42
4.10.2 Beschreibung der Konstruktoren und Destrukturen	42
4.10.2.1 SsK	42
4.10.2.2 SsK	42
4.10.2.3 ~SsK	42
4.10.3 Dokumentation der Elementfunktionen	42
4.10.3.1 bewerten	42
4.10.3.2 posSicher	43
4.10.4 Dokumentation der Datenelemente	43
4.10.4.1 gegner	43
4.10.4.2 gZuege	43
4.11 Stein Klassenreferenz	43
4.11.1 Ausführliche Beschreibung	46
4.11.2 Beschreibung der Konstruktoren und Destrukturen	46
4.11.2.1 Stein	46
4.11.2.2 Stein	46
4.11.2.3 ~Stein	47
4.11.3 Dokumentation der Elementfunktionen	47
4.11.3.1 getGeffangen	47
4.11.3.2 getid	47
4.11.3.3 getMteam	47
4.11.3.4 getOrt	47
4.11.3.5 setFrei	47
4.11.3.6 setGeffangen	47
4.11.3.7 setOrt	47
4.11.3.8 ziehenach	47
4.11.3.9 zuege	48
4.11.4 Dokumentation der Datenelemente	48
4.11.4.1 gefangen	48
4.11.4.2 id	48
4.11.4.3 mteam	49
4.11.4.4 ort	49
4.12 Strategie Klassenreferenz	49
4.12.1 Ausführliche Beschreibung	51
4.12.2 Beschreibung der Konstruktoren und Destrukturen	51
4.12.2.1 Strategie	51
4.12.2.2 Strategie	51
4.12.2.3 ~Strategie	51
4.12.3 Dokumentation der Elementfunktionen	51
4.12.3.1 bewerten	51

4.12.3.2	getmZuege	52
4.12.3.3	getWert	52
4.12.3.4	getZuege	52
4.12.3.5	nexZug	52
4.12.4	Dokumentation der Datenelemente	52
4.12.4.1	aZuege	52
4.12.4.2	brett	52
4.12.4.3	h1	52
4.12.4.4	h2	52
4.12.4.5	h3	52
4.12.4.6	k	52
4.12.4.7	mZuege	52
4.12.4.8	nZug	52
4.12.4.9	team	52
4.12.4.10	wert	53
4.13	Team Klassenreferenz	53
4.13.1	Ausführliche Beschreibung	55
4.13.2	Beschreibung der Konstruktoren und Destruktoren	55
4.13.2.1	Team	55
4.13.2.2	Team	55
4.13.2.3	~Team	55
4.13.3	Dokumentation der Elementfunktionen	55
4.13.3.1	distanzen	55
4.13.3.2	getBrett	56
4.13.3.3	getFarbe	56
4.13.3.4	getGegner	56
4.13.3.5	getSieg	56
4.13.3.6	getStein	56
4.13.3.7	setGegner	56
4.13.3.8	setSieg	56
4.13.4	Dokumentation der Datenelemente	56
4.13.4.1	brett	56
4.13.4.2	Farbe	57
4.13.4.3	gegner	57
4.13.4.4	helfer1	57
4.13.4.5	helfer2	57
4.13.4.6	helfer3	57
4.13.4.7	koenig	57
4.13.4.8	Sieg	57
4.14	User Klassenreferenz	57

4.14.1	Ausführliche Beschreibung	57
4.14.2	Beschreibung der Konstruktoren und Destruktoren	57
4.14.2.1	User	58
4.14.2.2	~User	58
4.14.3	Dokumentation der Elementfunktionen	58
4.14.3.1	Graphik	58
4.15	zug Strukturreferenz	58
4.15.1	Ausführliche Beschreibung	60
4.15.2	Beschreibung der Konstruktoren und Destruktoren	60
4.15.2.1	zug	60
4.15.2.2	zug	60
4.15.2.3	zug	60
4.15.3	Dokumentation der Elementfunktionen	60
4.15.3.1	operator<	60
4.15.3.2	operator=	60
4.15.3.3	operator==	60
4.15.4	Dokumentation der Datenelemente	60
4.15.4.1	stein	60
4.15.4.2	wert	60
4.15.4.3	zpos	60
4.15.4.4	zu	61
5	Datei-Dokumentation	63
5.1	Feld.cpp-Dateireferenz	63
5.1.1	Makro-Dokumentation	63
5.1.1.1	STEIN_C	63
5.2	Feld.h-Dateireferenz	64
5.3	GUI.cpp-Dateireferenz	64
5.4	GUI.h-Dateireferenz	65
5.5	KI.cpp-Dateireferenz	67
5.6	KI.h-Dateireferenz	68
5.7	Koenig.cpp-Dateireferenz	70
5.7.1	Makro-Dokumentation	71
5.7.1.1	KOEING_C	71
5.8	Koenig.h-Dateireferenz	71
5.8.1	Makro-Dokumentation	72
5.8.1.1	KOENIG_H	72
5.9	main.cpp-Dateireferenz	72
5.9.1	Dokumentation der Funktionen	73
5.9.1.1	main	74

5.10	Main.h-Dateireferenz	74
5.11	Possition.h-Dateireferenz	74
5.11.1	Makro-Dokumentation	74
5.11.1.1	POSSITION_H	74
5.12	SfH.cpp-Dateireferenz	75
5.13	SfH.h-Dateireferenz	76
5.14	SfK.cpp-Dateireferenz	77
5.15	SfK.h-Dateireferenz	78
5.16	SpielBrett.cpp-Dateireferenz	80
5.16.1	Makro-Dokumentation	81
5.16.1.1	SPIELBRETT_C	81
5.17	SpielBrett.h-Dateireferenz	81
5.18	SrH.cpp-Dateireferenz	82
5.19	SrH.h-Dateireferenz	83
5.20	SsK.cpp-Dateireferenz	85
5.21	SsK.h-Dateireferenz	85
5.22	Stein.cpp-Dateireferenz	87
5.22.1	Makro-Dokumentation	88
5.22.1.1	STEIN_C	88
5.23	Stein.h-Dateireferenz	88
5.24	Strategie.cpp-Dateireferenz	89
5.25	Strategie.h-Dateireferenz	90
5.26	Team.cpp-Dateireferenz	92
5.26.1	Makro-Dokumentation	92
5.26.1.1	TEAM_C	92
5.27	Team.h-Dateireferenz	92
5.28	User.cpp-Dateireferenz	94
5.28.1	Makro-Dokumentation	94
5.28.1.1	USER_C	94
5.29	User.h-Dateireferenz	94
5.30	zug.h-Dateireferenz	95
Index		97

Kapitel 1

Hierarchie-Verzeichnis

1.1 Klassenhierarchie

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

Feld	7
GUI	10
KI	13
Possition	23
SpielBrett	33
Stein	43
Koenig	20
Strategie	49
SfH	24
SfK	28
SrH	36
SsK	40
Team	53
User	57
zug	58

Kapitel 2

Klassen-Verzeichnis

2.1 Auflistung der Klassen

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

Feld	7
GUI	10
KI	13
Koenig	20
Possition	23
SfH	24
SfK	28
SpielBrett	33
SrH	36
SsK	40
Stein	43
Strategie	49
Team	53
User	57
zug	58

Kapitel 3

Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

Feld.cpp	63
Feld.h	64
GUI.cpp	64
GUI.h	65
KI.cpp	67
KI.h	68
Koenig.cpp	70
Koenig.h	71
main.cpp	72
Main.h	74
Possition.h	74
SfH.cpp	75
SfH.h	76
SfK.cpp	77
SfK.h	78
SpielBrett.cpp	80
SpielBrett.h	81
SrH.cpp	82
SrH.h	83
SsK.cpp	85
SsK.h	85
Stein.cpp	87
Stein.h	88
Strategie.cpp	89
Strategie.h	90
Team.cpp	92
Team.h	92
User.cpp	94
User.h	94
zug.h	95

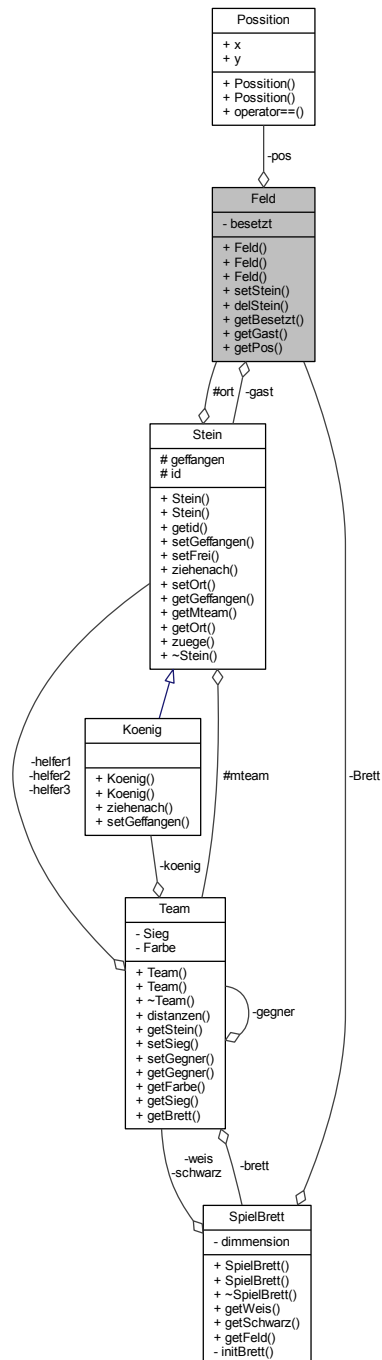
Kapitel 4

Klassen-Dokumentation

4.1 Feld Klassenreferenz

```
#include <Feld.h>
```

Zusammengehörigkeiten von Feld:



Öffentliche Methoden

- **Feld** ()
- **Feld** (short nx, short ny)
- **Feld** (**Feld** &f)
- void **setStein** (**Stein** *newstein)
- void **delStein** ()

- `bool getBesetzt ()`
- `Stein * getGast ()`
- `Possition getPos ()`

Private Attribute

- `bool besetzt`
- `Possition pos`
- `Stein * gast = nullptr`

4.1.1 Ausführliche Beschreibung

class `Feld` Diese Klasse Symbolisiert ein `Feld` auf einem Spielbrett.

4.1.2 Beschreibung der Konstruktoren und Destruktoren

4.1.2.1 `Feld::Feld ()`

4.1.2.2 `Feld::Feld (short nx, short ny)`

`Feld` Konstruktor

Parameter

<code>in</code>	<code>nx</code>	x Koordinaten des Feldes
<code>in</code>	<code>ny</code>	y Koordinaten des Feldes

4.1.2.3 `Feld::Feld (Feld & f)`

4.1.3 Dokumentation der Elementfunktionen

4.1.3.1 `void Feld::delStein ()`

`delStein` Loescht Zeiger auf Gast Setzt `besetzt` auf false

4.1.3.2 `bool Feld::getBesetzt ()`

Get the value of `besetzt`.

Rückgabe

the value of `besetzt`.

4.1.3.3 `Stein * Feld::getGast ()`

`getGast`

Rückgabe

Gibt einen Pointer auf den Gast zurueck.

4.1.3.4 Position `Feld::getPos ()`

`getPos`

Rückgabe

the value of pos.

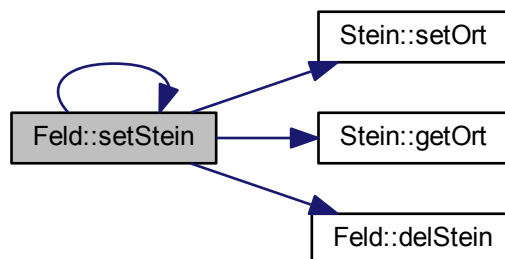
4.1.3.5 `void Feld::setStein (Stein * newstein)`

Setzt [Stein](#) auf das [Feld](#) und Markiert das [Feld](#) als Besetzt. Falls das [Feld](#) besetzt ist, werden die Gäste/Steine getauscht.

Parameter

<i>[in/out]</i>	*newstein pointer auf den zu setzenden Stein .
-----------------	--

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.1.4 Dokumentation der Datenelemente

4.1.4.1 `bool Feld::besetzt [private]`

4.1.4.2 `Stein* Feld::gast = nullptr [private]`

4.1.4.3 `Position Feld::pos [private]`

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [Feld.h](#)
- [Feld.cpp](#)

4.2 GUI Klassenreferenz

```
#include <GUI.h>
```

[illegible]

- GUI (SpielBrett *br, KI *ki)
- void zeichneSpielfeld (int zug, int spieler)
- void zeichneAnleitung ()
- void zeichneZug (int zug, int spieler, int zeile, int spalte)
- void Spieler (bool farbe, int zug, int spieler)

Private Attribute

- SpielBrett * brett
- KI * KlsW

4.2.1 Ausführliche Beschreibung

class GUI Die Abkürzung GUI steht für graphical user interface bzw. grafische Benutzeroberfläche. Durch diese Klasse wird dem Benutzer eine grafische Oberfläche zur Verfügung gestellt, über die er mit dem Programm interagieren kann. Alle Benutzereingaben erfolgen ausschließlich über die Tastatur. Unterstützend wird die Struktur der erwarteten Eingabe in Klammern mit angegeben. Sollte dennoch der Benutzer eine Falsch Eingabe tätigen, so wird er darauf hingewiesen und kann seine Eingabe nach 3 Sekunden wiederholen. Die Darstellung des Spielfeldes und wichtiger Spielparameter erfolgt in der Windows Konsole über ANSI-Zeichen.

4.2.2 Beschreibung der Konstruktoren und Destruktoren

4.2.2.1 GUI::GUI (SpielBrett * br, KI * ki)

4.2.3 Dokumentation der Elementfunktionen

4.2.3.1 void GUI::Spieler (bool farbe, int zug, int spieler)

zeichneZug(zug,spieler,zeile,spalte) Mit dieser Funktion wird dem Spieler alle zulässigen Züge des ausgewählten Steins angezeigt.

Parameter

[int]	zug gibt den aktuellen Zug an
[int]	spieler gibt an, welcher Spieler gerade am Zug ist
[int]	zeile Zeile des ausgewählten Steins
[int]	spalte Spalte des ausgewählten Steins

4.2.3.2 void GUI::zeichneAnleitung ()

zeichneSpielfeld(zug,spieler) Mit dieser Funktion wird das Spielfeld grafisch für den Spieler aufbereitet.

Parameter

[int]	zug gibt den aktuelle Zug an
[int]	spieler gibt an, welcher Spieler an Zug ist

4.2.3.3 void GUI::zeichneSpielfeld (int zug, int spieler)

GUI Diese Funktion ist ein Konstruktor für eine Instanz von der Klasse Spielbrett.

4.2.3.4 void GUI::zeichneZug (int zug, int spieler, int zeile, int spalte)

zeichenAnleitung() Diese Funktion gibt dem Benutzer Auskunft über die Spielregeln.

4.2.4 Dokumentation der Datenelemente

4.2.4.1 SpielBrett* GUI::brett [private]

4.2.4.2 KI* GUI::Klsw [private]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [GUI.h](#)

- [GUI.cpp](#)

4.3 KI Klassenreferenz

```
#include <KI.h>
```


Private Methoden

- `std::vector< zug > mergeStrategie (Strategie *st1, std::vector< zug > st2Zuege)`
- `std::vector< zug > mergeStrategie (Strategie *st1, Strategie *st2)`
- `void seachBestZug ()`

Private Attribute

- `Team & t`
- `SpielBrett & abrett`
- `Strategie * strat [anzstrat]`
- `SsK * dv`
- `zug nZug`

Statische, private Attribute

- `static const int anzstrat =4`

4.3.1 Ausführliche Beschreibung

class **KI** Ist eine Klasse die aus den möglichen Spielzügen den besten auswählt. Sie ist mit zusätzlichen Strategien erweiterbar.

4.3.2 Beschreibung der Konstruktoren und Destruktoren

4.3.2.1 KI::KI (Team & t)

KI Konstruktor

Parameter

<code>in, out</code>	<code>t</code>	Referenz auf das Team , das gesteuert werden soll
----------------------	----------------	--

4.3.2.2 KI::~KI () [virtual]

4.3.3 Dokumentation der Elementfunktionen

4.3.3.1 SpielBrett & KI::getBrett ()

getBrett

Rückgabe

Referenz auf das Spielbrett

4.3.3.2 Team & KI::getTeam ()

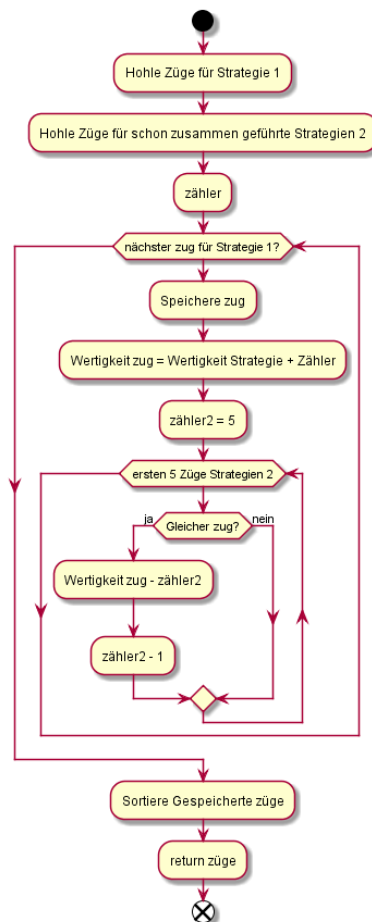
getTeam

Rückgabe

Referenz auf das gesteuerte **Team**

4.3.3.3 `std::vector< zug > KI::mergeStrategie (Strategie * st1, std::vector< zug > st2Zuege)` [private]

`mergeStrategie` Vereint zwei Strategien und führt die Wertigkeiten zusammen. Je kleiner die Wertigkeits-Zahl desto besser ist der zug.



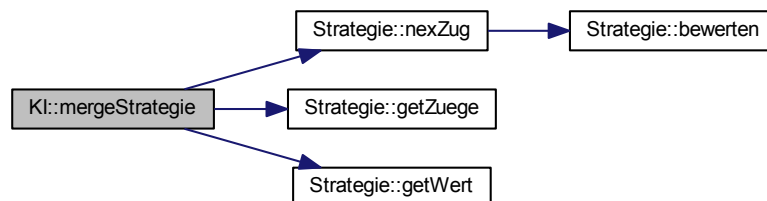
Parameter

in	<code>st1</code>	Pointer auf eine Strategie
in	<code>st2Zuege</code>	Vector mit Zügen.

Rückgabe

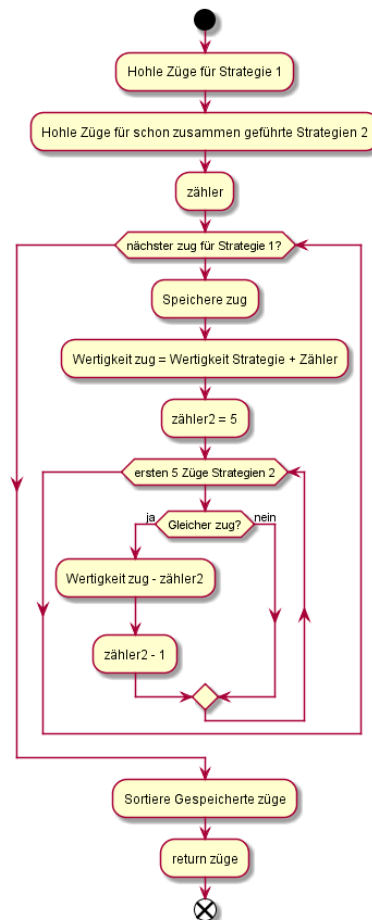
Vector mit nach wertigkeit sortierten zügen.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.3.3.4 std::vector< zug > KI::mergeStrategie (Strategie * st1, Strategie * st2) [private]

- mergeStrategie Vereint zwei Strategien und führt die Wertigkeiten zusammen. Je kleiner die wertigkeits Zahl desto besser ist der Zug.



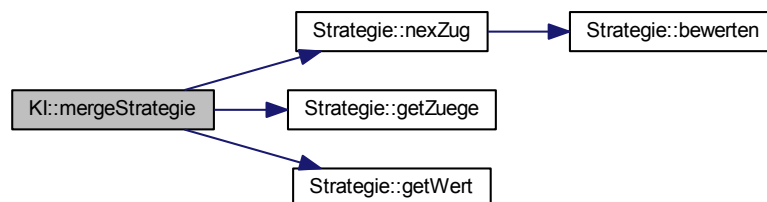
Parameter

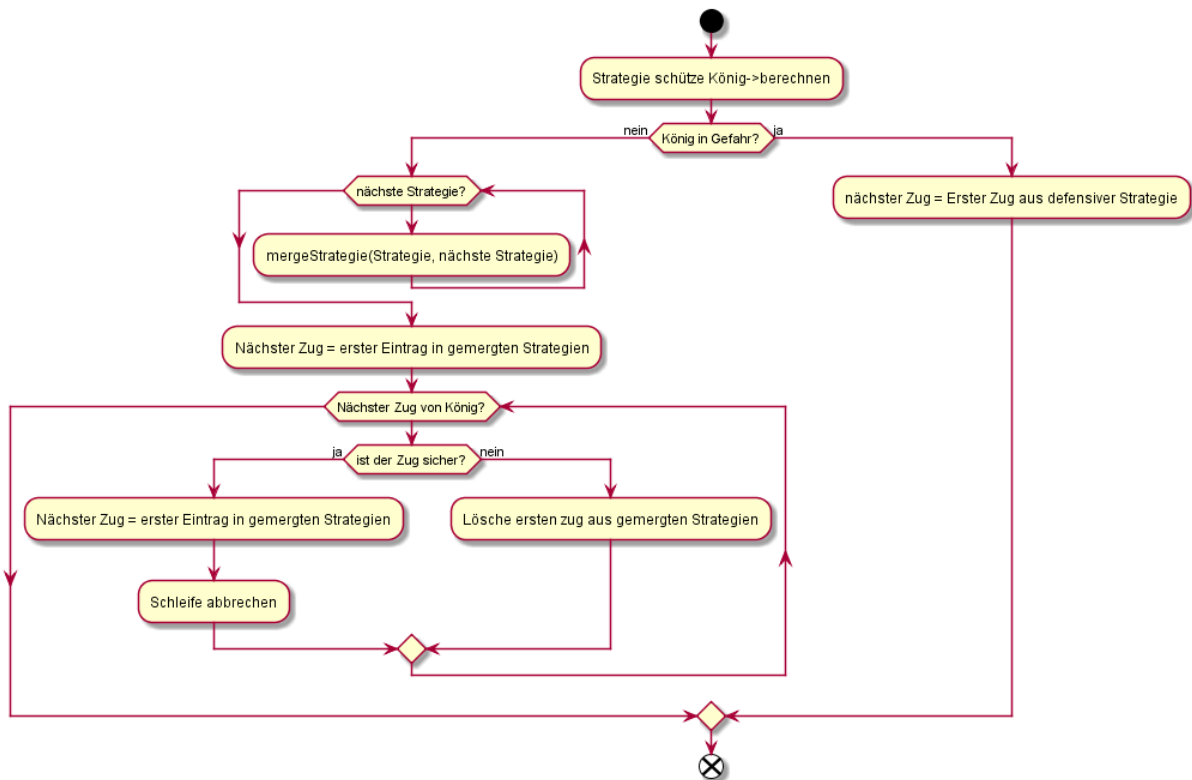
in	st1	Pointer auf eine Strategie .
in	st2	Pointer auf eine Strategie .

Rückgabe

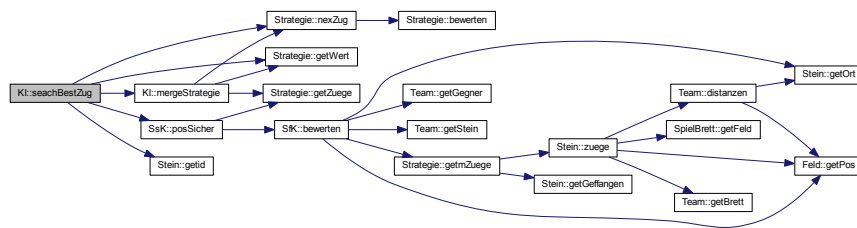
Vector mit nach Wertigkeit sortierten Zügen.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:





Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.3.4 Dokumentation der Datenelemente

4.3.4.1 **SpielBrett**& **KI::abrett** [private]

4.3.4.2 **const int KI::anzstrat=4** [static],[private]

4.3.4.3 **SsK*** **KI::dv** [private]

4.3.4.4 **zug** **KI::nZug** [private]

4.3.4.5 **Strategie*** **KI::strat[anzstrat]** [private]

4.3.4.6 **Team**& **KI::t** [private]

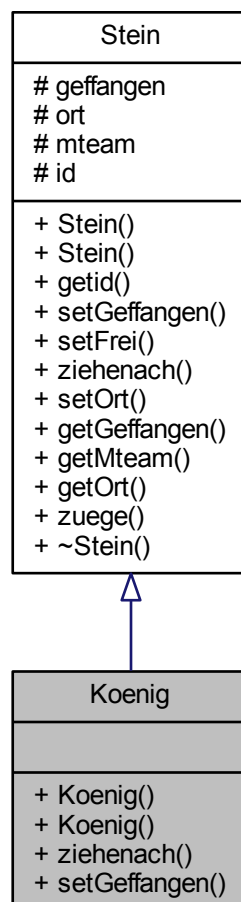
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [KI.h](#)
- [KI.cpp](#)

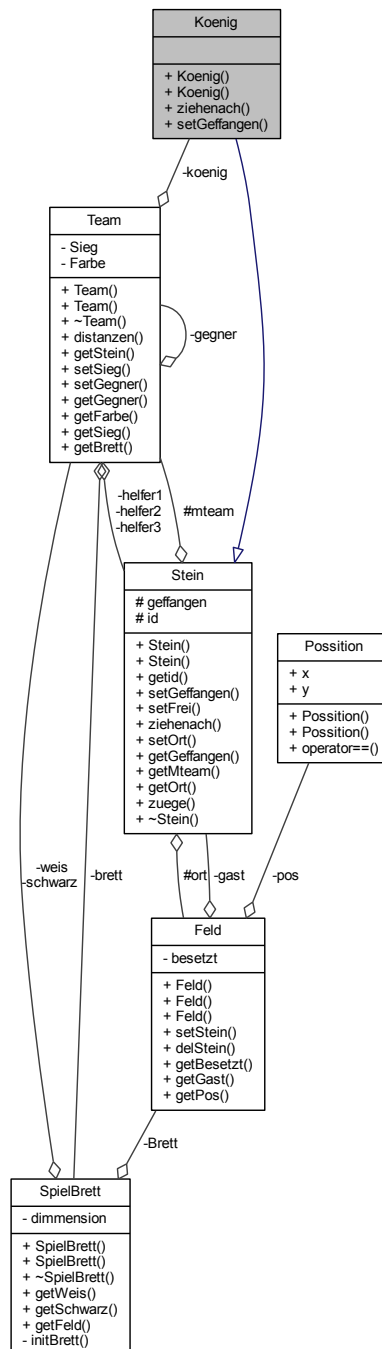
4.4 Koenig Klassenreferenz

```
#include <Koenig.h>
```

Klassendiagramm für Koenig:



Zusammengehörigkeiten von Koenig:



Öffentliche Methoden

- [Koenig](#) ()
- [Koenig](#) (int id, [Feld](#) *startplatz, [Team](#) *mt)
- virtual bool [ziehenach](#) ([Feld](#) *ziehe) override
- virtual void [setGefangen](#) () override

Weitere Geerbte Elemente

4.4.1 Beschreibung der Konstruktoren und Destruktoren

4.4.1.1 `Koenig::Koenig ()`

4.4.1.2 `Koenig::Koenig (int id, Feld * startplatz, Team * mt)`

4.4.2 Dokumentation der Elementfunktionen

4.4.2.1 `void Koenig::setGeffangen ()` `[override],[virtual]`

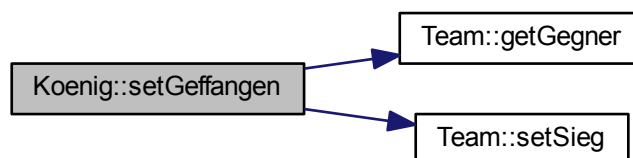
`ziehenach(Feld)` Mit dieser Methode hat der König die Möglichkeit seine Helfer zu befreien und sich auf dem Spielfeld zu bewegen.

Parameter

<code>[Feld]</code>	*ziehe hierbei handelt sich um einen Pointer auf das Feld , das er springen soll
---------------------	--

Erneute Implementation von [Stein](#).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

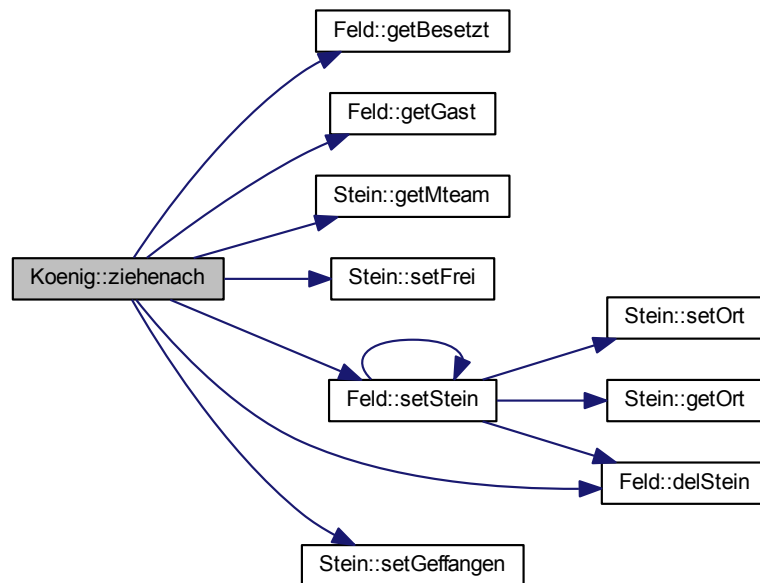


4.4.2.2 `bool Koenig::ziehenach (Feld * ziehe)` `[override],[virtual]`

Implementiert den Startplatz des Königs.

Erneute Implementation von [Stein](#).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [Koenig.h](#)
- [Koenig.cpp](#)

4.5 Possition Strukturreferenz

```
#include <Possition.h>
```

Zusammengehörigkeiten von Possition:

Possition
+ x + y
+ Possition() + Possition() + operator==()

Öffentliche Methoden

- [Position](#) (short int [x](#), short int [y](#))
- [Position](#) ()
- bool [operator==](#) (const [Position](#) &p) const

Öffentliche Attribute

- short int [x](#)
- short int [y](#)

4.5.1 Ausführliche Beschreibung

struct Position

4.5.2 Beschreibung der Konstruktoren und Destruktoren

4.5.2.1 [Position::Position](#) (short int [x](#), short int [y](#)) `[inline]`

4.5.2.2 [Position::Position](#) () `[inline]`

4.5.3 Dokumentation der Elementfunktionen

4.5.3.1 bool [Position::operator==](#) (const [Position](#) & [p](#)) const `[inline]`

4.5.4 Dokumentation der Datenelemente

4.5.4.1 short int [Position::x](#)

4.5.4.2 short int [Position::y](#)

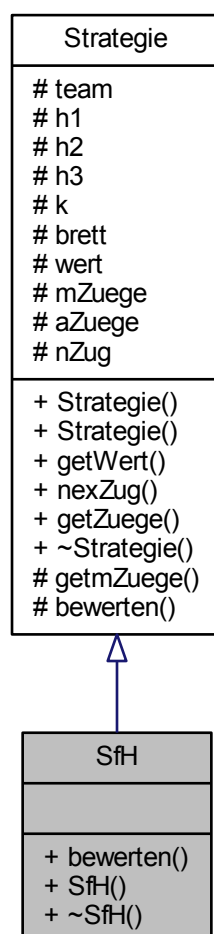
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [Position.h](#)

4.6 SfH Klassenreferenz

```
#include <SfH.h>
```

Klassendiagramm für SfH:



Weitere Geerbte Elemente

4.6.1 Ausführliche Beschreibung

class [SfH](#) ([Strategie](#) fange Helfer) implementiert die Methode [bewerten\(\)](#);

Diese [Strategie](#) sorgt dafür, dass die gegnerischen Helfer festgesetzt/gefangen werden. Ein festgesetzter/gefangener Helfer stellt insofern keine Bedrohung mehr dar, bis er wieder vom [Koenig](#) befreit wird. Dies gilt es durch andere Strategien zu verhindern.

Parameter

<i>&team</i>	Referenz auf Instanz von Team
<i>&b</i>	Referenz auf Instanz von SpielBrett

4.6.2 Beschreibung der Konstruktoren und Destruktoren

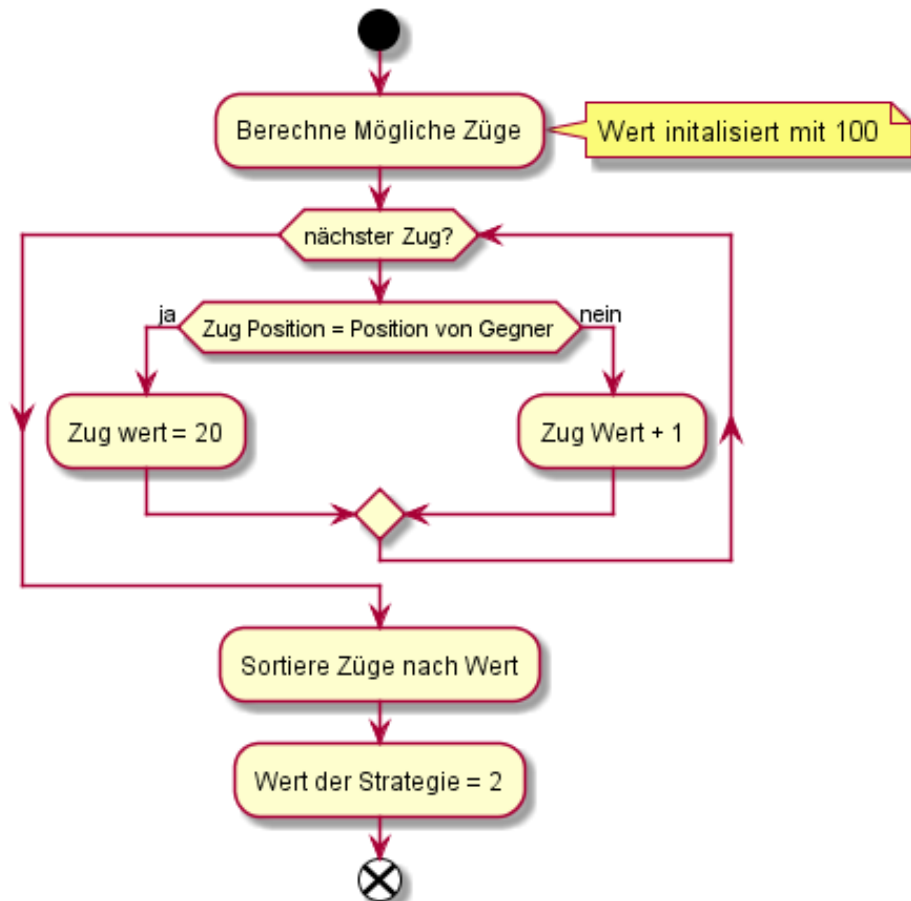
4.6.2.1 `SfH::SfH (Team & team, SpielBrett & b)`

4.6.2.2 `SfH::~~SfH () [virtual]`

4.6.3 Dokumentation der Elementfunktionen

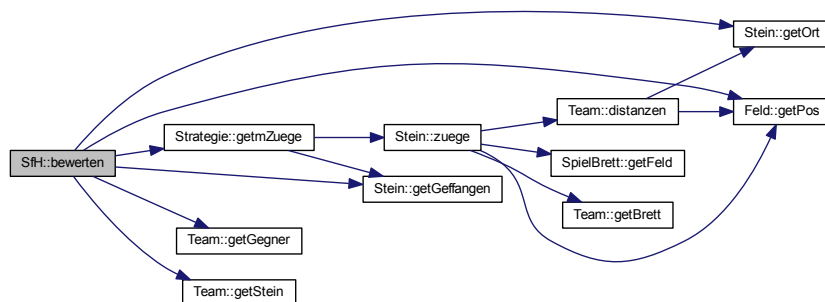
4.6.3.1 `void SfH::bewerten () [override],[virtual]`

[bewerten\(\)](#) Bewertet mögliche Züge nach der Möglichkeit gegnerische Helfer zu fangen.



Implementiert [Strategie](#).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



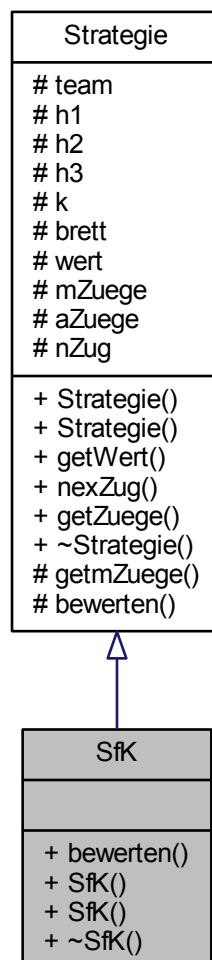
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [SfH.h](#)
- [SfH.cpp](#)

4.7 SfK Klassenreferenz

```
#include <SfK.h>
```

Klassendiagramm für SfK:



Weitere Geerbte Elemente

4.7.1 Ausführliche Beschreibung

class [SfK](#) ([Strategie](#) fange König) Ist eine Ableitung der abstrakten Klasse [Strategie](#).

Diese [Strategie](#) sorgt dafür, dass sich die Spielfiguren dem gegnerischen König nähern, um in festsetzen/gefangen nehmen zu können. Ein festgesetzter/gefangener König bedeutet das Spielende. Ein Sieg wird erzielt, sobald der gegnerische König festgesetzt/gefangen ist.

Überschreibt/implementiert die Methode [bewerten\(\)](#);

Parameter

<i>&team</i>	Referenz auf Instanz von Team
<i>&b</i>	Referenz auf Instanz von SpielBrett

4.7.2 Beschreibung der Konstruktoren und Destruktoren

4.7.2.1 `SfK::SfK (Team & team, SpielBrett & b)`

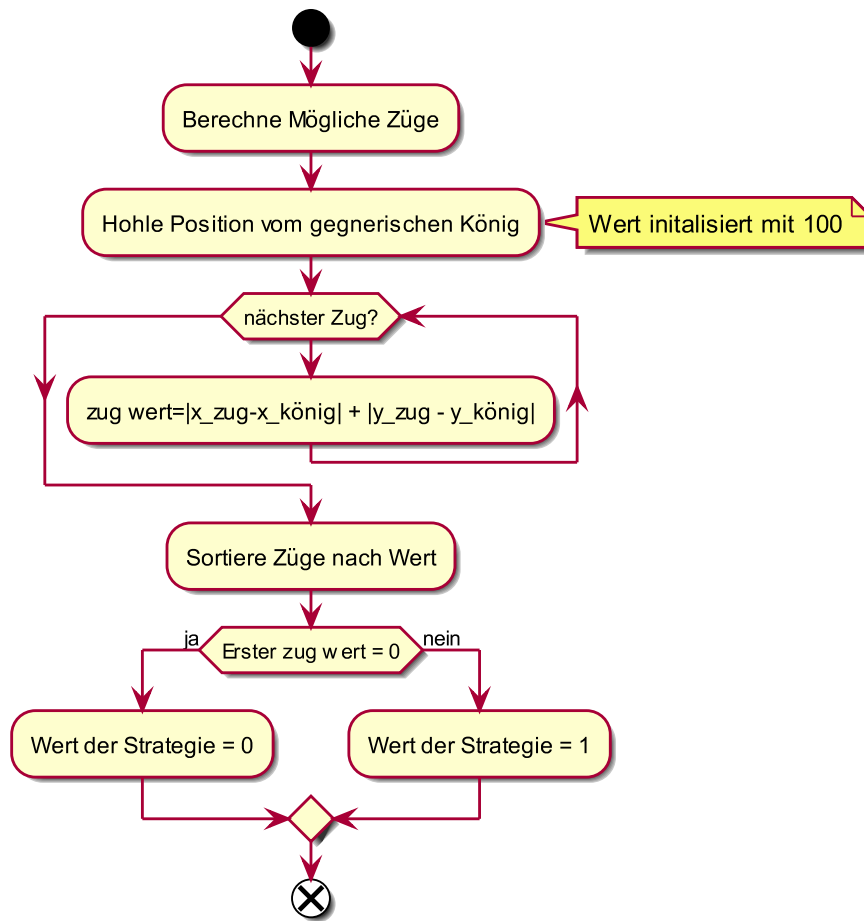
4.7.2.2 `SfK::SfK ()`

4.7.2.3 `SfK::~~SfK () [virtual]`

4.7.3 Dokumentation der Elementfunktionen

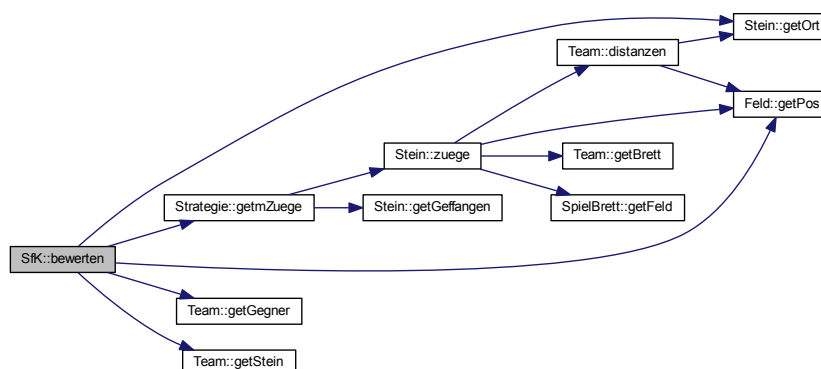
4.7.3.1 `void SfK::bewerten () [override],[virtual]`

[bewerten\(\)](#) Bewertet mögliche Züge nach der Möglichkeit gegnerischen König zu fangen.



Implementiert [Strategie](#).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



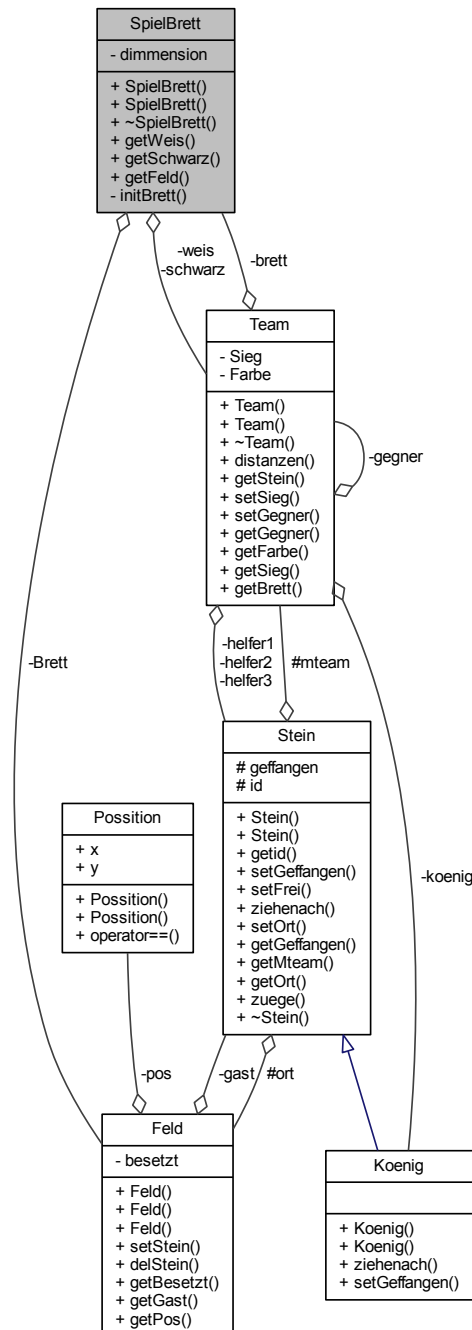
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [SfK.h](#)
- [SfK.cpp](#)

4.8 SpielBrett Klassenreferenz

```
#include <SpielBrett.h>
```

Zusammengehörigkeiten von SpielBrett:



Öffentliche Methoden

- [SpielBrett](#) ()
- [SpielBrett](#) (const [SpielBrett](#) &sb)

- `~SpielBrett()`
- `Team * getWeis()` const
- `Team * getSchwarz()` const
- `Feld * getFeld(int x, int y)` const

Private Methoden

- void `initBrett()`

Private Attribute

- `Feld ** Brett = nullptr`
- `Team * schwarz = nullptr`
- `Team * weis = nullptr`

Statische, private Attribute

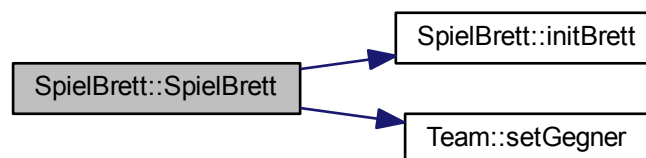
- static const short int `dimension = 8`

4.8.1 Beschreibung der Konstruktoren und Destruktoren

4.8.1.1 `SpielBrett::SpielBrett()`

`initBrett()` Erzeugt das 8x8 großes `Feld`.

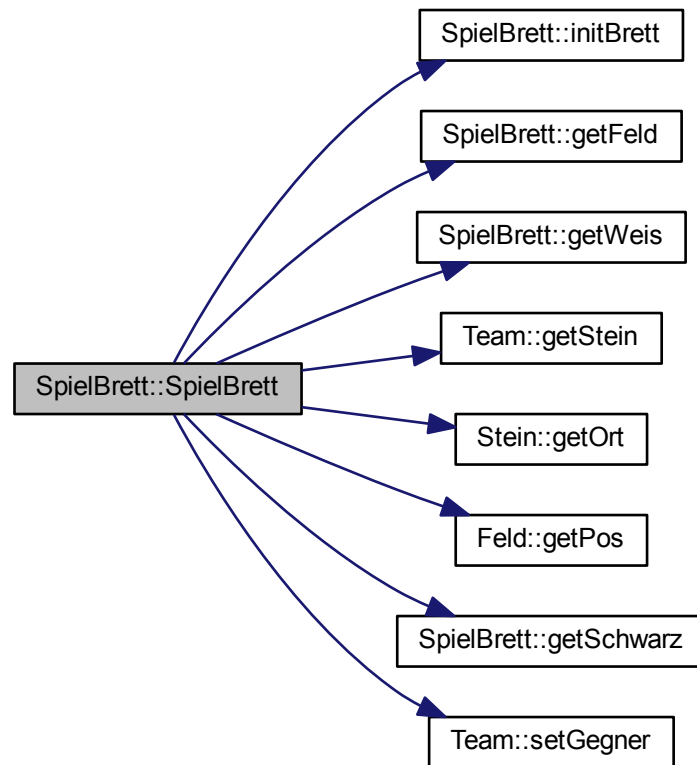
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.8.1.2 `SpielBrett::SpielBrett(const SpielBrett & sb)`

`SpielBrett()` Beinhaltet die Startaufstellung.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.8.1.3 SpielBrett::~~SpielBrett ()

`SpielBrett` (const `SpielBrett` &sb) Verweist auf die Pointer, der einzelnen Spielsteine.

4.8.2 Dokumentation der Elementfunktionen

4.8.2.1 Feld * SpielBrett::getFeld (int x, int y) const

`getSchwarz()` Kennzeichnet die schwarzen Steine.

Rückgabe

schwarz

4.8.2.2 Team * SpielBrett::getSchwarz () const

`getWeis()` Kennzeichnet die weißen Steine.

Rückgabe

weiß

4.8.2.3 **Team * SpielBrett::getWeis () const**

~SpielBrett Destruktor des Spiels

4.8.2.4 **void SpielBrett::initBrett () [private]**

4.8.3 Dokumentation der Datenelemente

4.8.3.1 **Feld** SpielBrett::Brett =nullptr [private]**

4.8.3.2 **const short int SpielBrett::dimension = 8 [static],[private]**

4.8.3.3 **Team* SpielBrett::schwarz =nullptr [private]**

4.8.3.4 **Team * SpielBrett::weis =nullptr [private]**

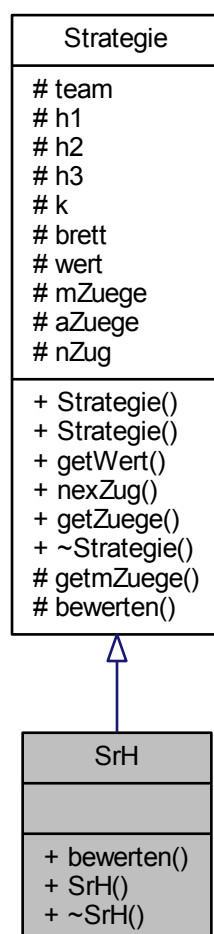
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [SpielBrett.h](#)
- [SpielBrett.cpp](#)

4.9 SrH Klassenreferenz

```
#include <SrH.h>
```

Klassendiagramm für SrH:



Weitere Geerbte Elemente

4.9.1 Ausführliche Beschreibung

class [SrH](#) ([Strategie](#) rette Helfer) Ist eine Ableitung der abstrakten Klasse [Strategie](#).

Diese [Strategie](#) sorgt dafür, dass der [Koenig](#) teameigene festgesetzte/gefangene Helfer befreit. Dies tut er allerdings nach Möglichkeit erst dann, wenn sie sich auch in unmittelbarer Umgebung befinden, da der [Koenig](#) selber eine sehr defensive Rolle im Spielverlauf einnimmt.

Ueberschreibt/implementiert die Methode [bewerten\(\)](#);

Parameter

<i>&team</i>	Referenz auf Instanz von Team
<i>&b</i>	Referenz auf Instanz von SpielBrett

4.9.2 Beschreibung der Konstruktoren und Destruktoren

4.9.2.1 [SrH::SrH](#) ([Team](#) & *team*, [SpielBrett](#) & *b*)

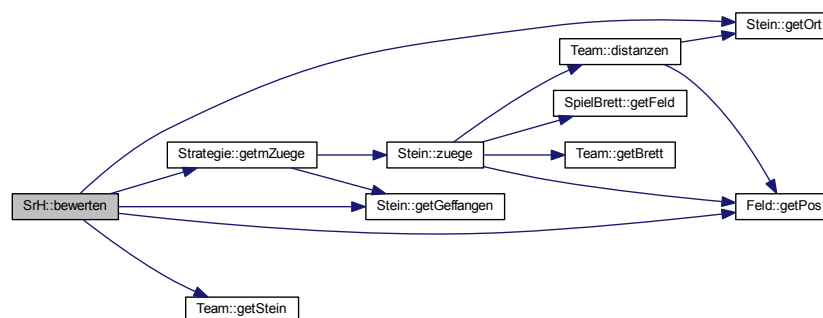
4.9.2.2 [SrH::~~SrH](#) () [virtual]

4.9.3 Dokumentation der Elementfunktionen

4.9.3.1 [void SrH::bewerten](#) () [override],[virtual]

Implementiert [Strategie](#).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



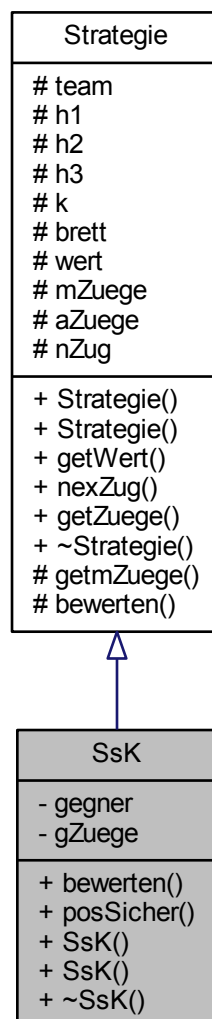
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [SrH.h](#)
- [SrH.cpp](#)

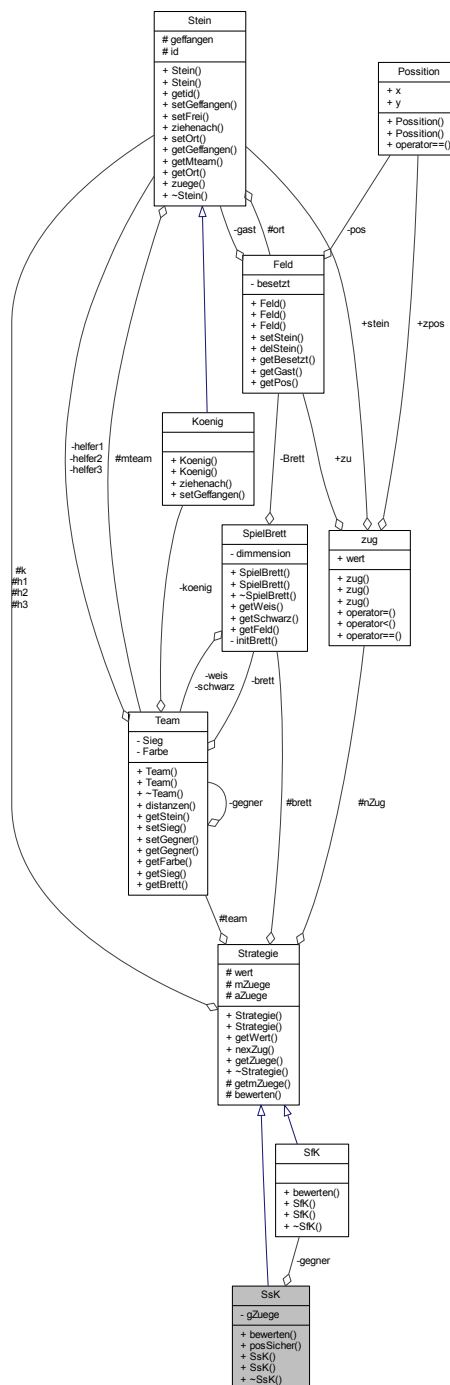
4.10 SsK Klassenreferenz

```
#include <SsK.h>
```

Klassendiagramm für SsK:



Zusammengehörigkeiten von SsK:



Öffentliche Methoden

- virtual void **bewerten** () override
- bool **posSicher** (Position p)
- **SsK** (Team &team, SpielBrett &b)
- **SsK** ()
- virtual **~SsK** ()

Private Attribute

- `SfK` `gegner`
- `std::vector< zug >` `gZuege`

Weitere Geerbte Elemente

4.10.1 Ausführliche Beschreibung

class `SsK` (`Strategie` schuetze `Koenig`) Ist eine Ableitung der abstrakten Klasse `Strategie`.

Diese `Strategie` sorgt dafuer, dass der teameigene `Koenig` vor festsetzen/gefangen nehmen durch feindliche Spielfiguren geschuetzt wird. Zu beobachten ist hierbei das fangen von gegnerischen Spielfiguren, sobald sie dem König zu nahe kommen. Auch der `Koenig` selber nimmt ein sehr defensives Verhalten an und haelt sich von den Gegnern fern, um ein fruehzeitiges Ableben zu verhindern.

Ueberschreibt/implementiert die Methode `bewerten()`;

Parameter

<code>&team</code>	Referenz auf Instanz von <code>Team</code>
<code>&b</code>	Referenz auf Instanz von <code>SpielBrett</code>

4.10.2 Beschreibung der Konstruktoren und Destruktoren

4.10.2.1 `SsK::SsK (Team & team, SpielBrett & b)`

4.10.2.2 `SsK::SsK ()`

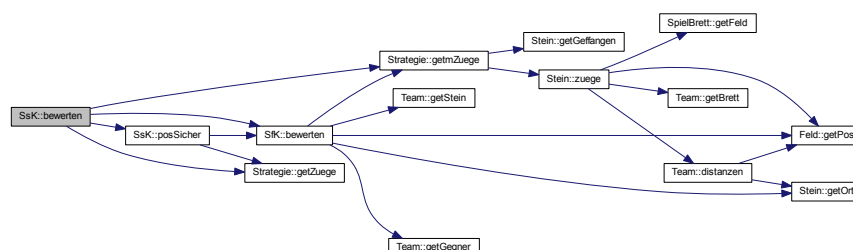
4.10.2.3 `SsK::~~SsK ()` `[virtual]`

4.10.3 Dokumentation der Elementfunktionen

4.10.3.1 `void SsK::bewerten ()` `[override]`, `[virtual]`

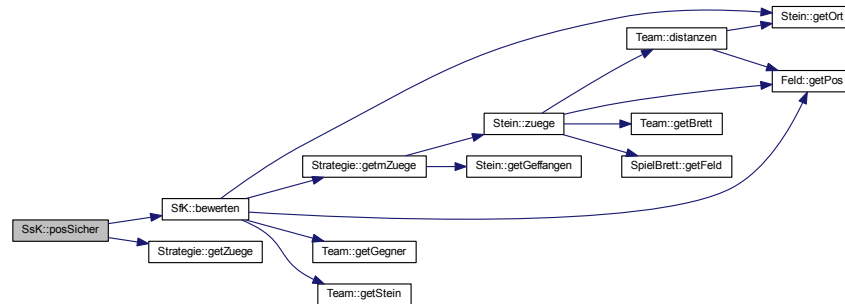
Implementiert `Strategie`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.10.3.2 `bool SsK::posSicher (Position p)`

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.10.4 Dokumentation der Datenelemente

4.10.4.1 `SfK SsK::gegner` [private]4.10.4.2 `std::vector<zug> SsK::gZuege` [private]

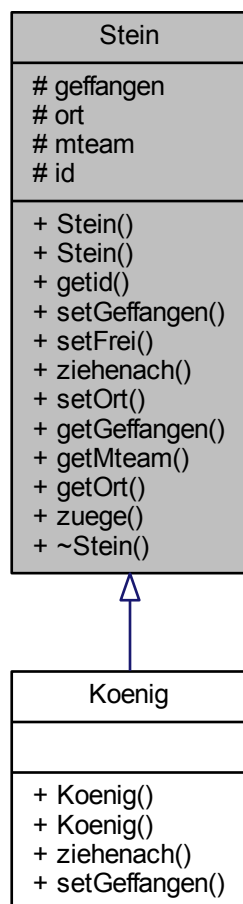
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [SsK.h](#)
- [SsK.cpp](#)

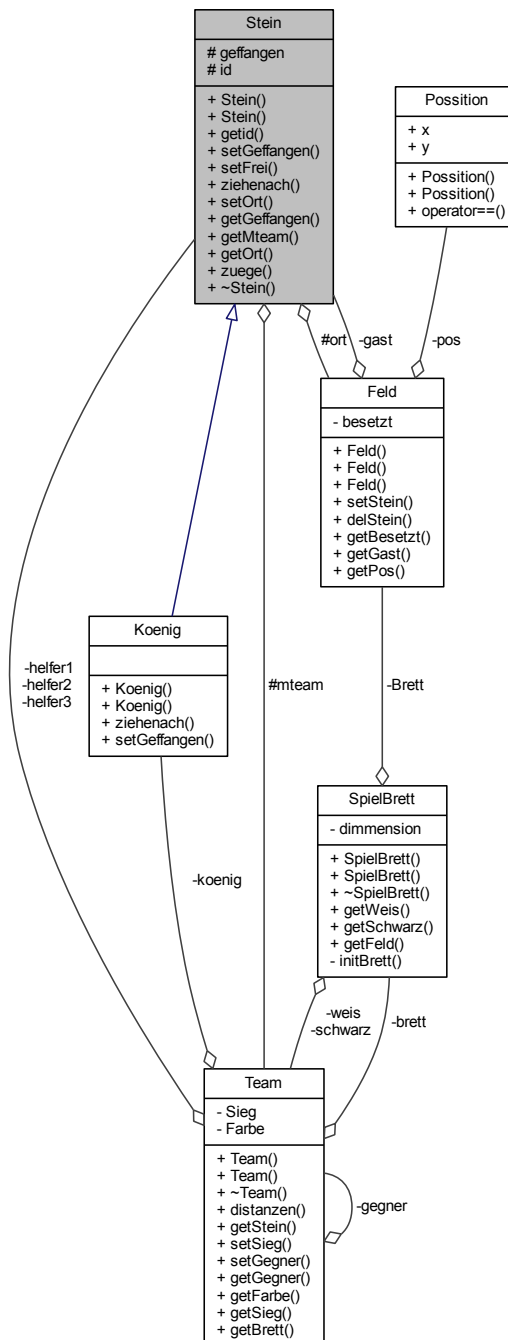
4.11 Stein Klassenreferenz

```
#include <Stein.h>
```

Klassendiagramm für Stein:



Zusammengehörigkeiten von Stein:



Öffentliche Methoden

- `Stein ()`
- `Stein (int id, Feld *startplatz, Team *mt)`
- `int getId () const`
- `virtual void setGefangen ()`
- `void setFrei ()`

- virtual bool `ziehenach` (`Feld *ziehl`)
- void `setOrt` (`Feld *o`)
- bool `getGefangen` ()
- `Team * getMteam` ()
- `Feld * getOrt` ()
- `std::vector< Feld * > zuege` ()
- virtual `~Stein` ()=default

Geschützte Attribute

- bool `geffangen` =false
- `Feld * ort` =nullptr
- `Team * mteam` =nullptr
- const int `id`

4.11.1 Ausführliche Beschreibung

class `Stein`

Jedes `Team` besitzt drei Helfer. Sie können sich auf dem Spielfeld bewegen, festgesetzt (gefangen) werden, gegnerische Spielfiguren festsetzen, indem man sie ganz einfach auf das vom Gegner besetzte `Feld` schickt und in Verbindung mit dem teameigenen König können sie auch selber befreit werden, sollte der Gegner sie gefangen genommen haben. Jede Spielfigur und damit auch jeder Helfer, bekommt bei Spielbeginn einen Platz mittels Pointern zugewiesen. Die Spielfigur-ID und die Spielfeld-ID bestimmen also, welche Spielfigur von welchem `Team` sich wo im `Feld` befindet.

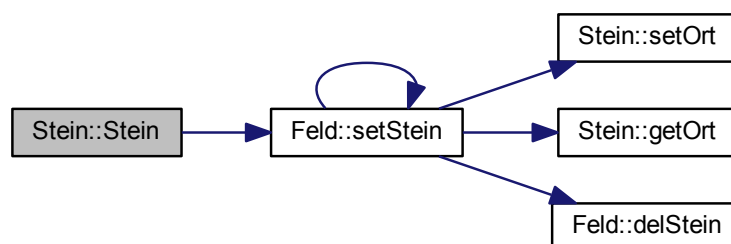
4.11.2 Beschreibung der Konstruktoren und Destruktoren

4.11.2.1 `Stein::Stein ()`

Konstruktor

4.11.2.2 `Stein::Stein (int id, Feld * startplatz, Team * mt)`

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.11.2.3 `virtual Stein::~~Stein () [virtual],[default]`

4.11.3 Dokumentation der Elementfunktionen

4.11.3.1 `bool Stein::getGefangen ()`

`getGefangen()` Die Funktion beschreibt, ob der [Stein](#) gefangen ist oder nicht.

Rückgabe

the value of gefangen

4.11.3.2 `int Stein::getid () const`

`getid()` `getid()` Diese Funktion sagt aus, ob es sich hierbei um weiß oder schwarz handelt.

Rückgabe

id der Instanz

4.11.3.3 `Team * Stein::getMteam ()`

4.11.3.4 `Feld * Stein::getOrt ()`

4.11.3.5 `void Stein::setFrei ()`

`setFrei()` Setzt den [Stein](#) frei Setzt gefangen -> false

4.11.3.6 `void Stein::setGefangen () [virtual]`

`setGefangen()` Setzt den [Stein](#) gefangen. gefangen -> true

Erneute Implementation in [Koenig](#).

4.11.3.7 `void Stein::setOrt (Feld * o)`

4.11.3.8 `bool Stein::ziehenach (Feld * ziehl) [virtual]`

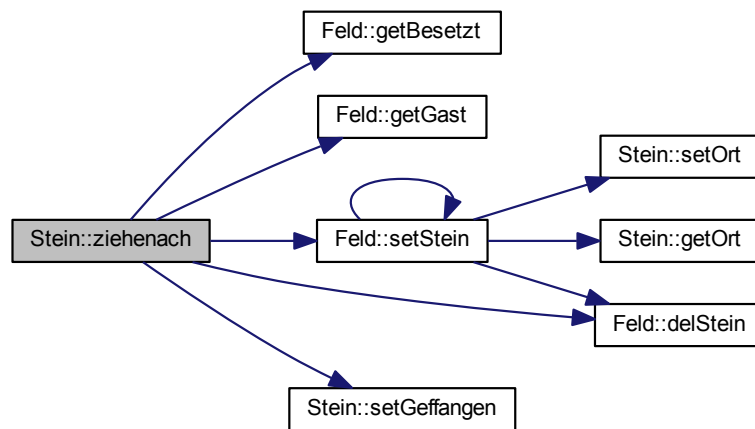
`setOrt` Rückt auf das übergebene [Feld](#).

Parameter

<code>[Feld]</code>	gibt die neue Position an
---------------------	---------------------------

Erneute Implementation in [Koenig](#).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



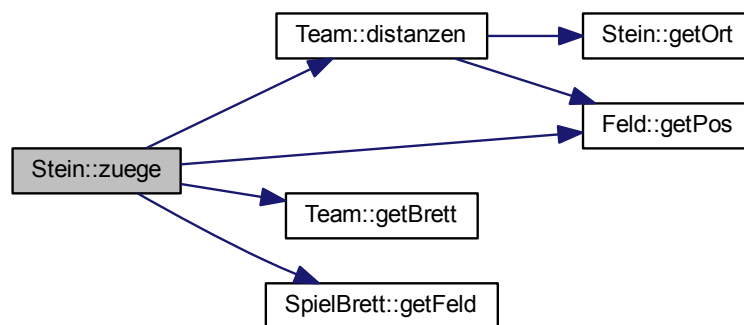
4.11.3.9 `std::vector< Feld * > Stein::zuege ()`

`zuege()` Die Funktion Zuege ermittelt alle möglichen Züge und gibt diese als Vector zurück.

Rückgabe

`Feld` zue

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.11.4 Dokumentation der Datenelemente

4.11.4.1 `bool Stein::geffangen = false` [protected]

4.11.4.2 `const int Stein::id` [protected]

4.11.4.3 `Team* Stein::mteam = nullptr` [protected]

4.11.4.4 `Feld* Stein::ort = nullptr` [protected]

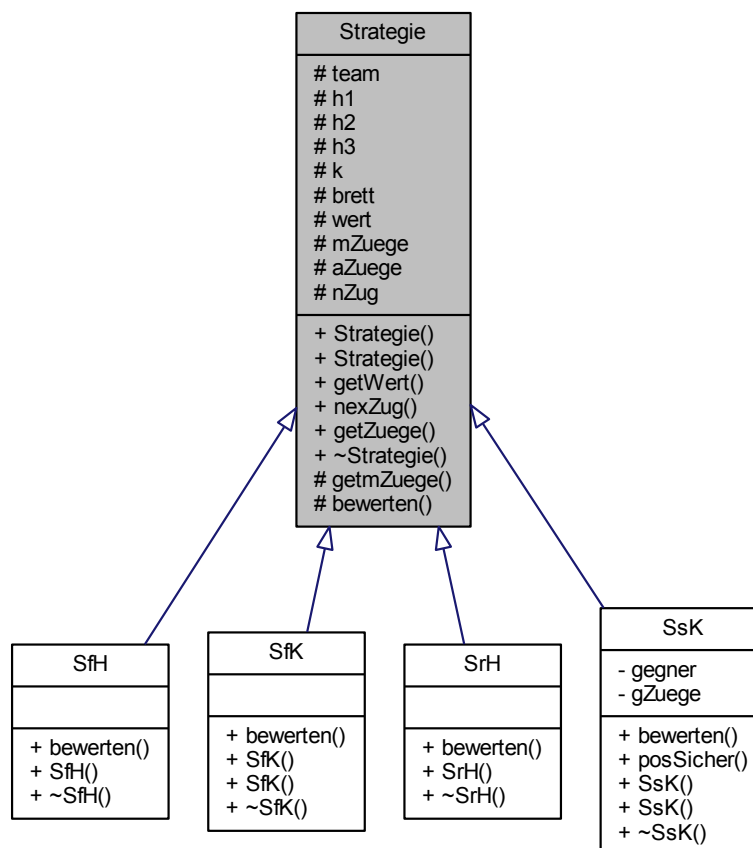
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [Stein.h](#)
- [Stein.cpp](#)

4.12 Strategie Klassenreferenz

```
#include <Strategie.h>
```

Klassendiagramm für Strategie:



Geschützte Methoden

- void `getmZuege` (std::vector< `zug` > &zuege)
- virtual void `bewerten` ()=0

Geschützte Attribute

- `Team` & `team`
- `Stein` & `h1`
- `Stein` & `h2`
- `Stein` & `h3`
- `Stein` & `k`
- `SpielBrett` & `brett`
- int `wert`
- std::vector< `zug` > `mZuege`
- std::vector< `zug` > `aZuege`
- `zug` `nZug`

4.12.1 Ausführliche Beschreibung

class `Strategie`

Abstrakte Klasse zur Erzeugung von speziellen Zug-Strategien.

Als Strategien sind jene Funktionen gemeint, welche neben der Bewegung im `Feld`, zusätzlich auch dafür sorgen, dass es zu einer Sieg/Niederlage Situation kommt. Sie stellen die Möglichkeiten dar, welche die Spielfiguren in den jeweiligen Momenten besitzen. Die Bewertung erfolgt in Echtzeit.

Wir programmierten 4 Strategien ein. Jede der 4 Strategien ist eine Vererbung dieser Klasse.

4.12.2 Beschreibung der Konstruktoren und Destruktoren

4.12.2.1 `Strategie::Strategie (Team & team, SpielBrett & b)`

4.12.2.2 `Strategie::Strategie ()`

4.12.2.3 `Strategie::~~Strategie ()` [virtual]

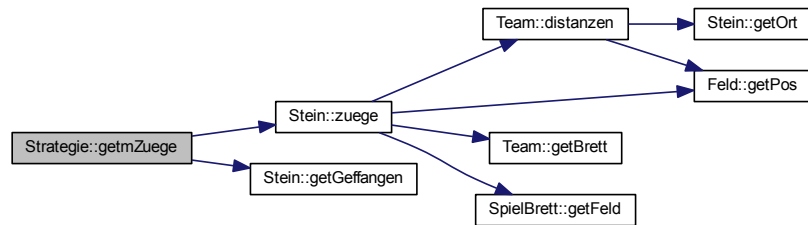
4.12.3 Dokumentation der Elementfunktionen

4.12.3.1 `void Strategie::bewerten ()` [protected],[pure virtual]

Implementiert in `SfK`, `SfH`, `SsK` und `SrH`.

4.12.3.2 void Strategie::getmZuege (std::vector< zug > & zuege) [protected]

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.12.3.3 int Strategie::getWert () const

4.12.3.4 std::vector< zug > Strategie::getZuege () const

4.12.3.5 zug Strategie::nexZug ()

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.12.4 Dokumentation der Datenelemente

4.12.4.1 std::vector<zug> Strategie::aZuege [protected]

4.12.4.2 SpielBrett& Strategie::brett [protected]

4.12.4.3 Stein& Strategie::h1 [protected]

4.12.4.4 Stein & Strategie::h2 [protected]

4.12.4.5 Stein & Strategie::h3 [protected]

4.12.4.6 Stein & Strategie::k [protected]

4.12.4.7 std::vector<zug> Strategie::mZuege [protected]

4.12.4.8 zug Strategie::nZug [protected]

4.12.4.9 Team& Strategie::team [protected]

4.12.4.10 `int Strategie::wert` `[protected]`

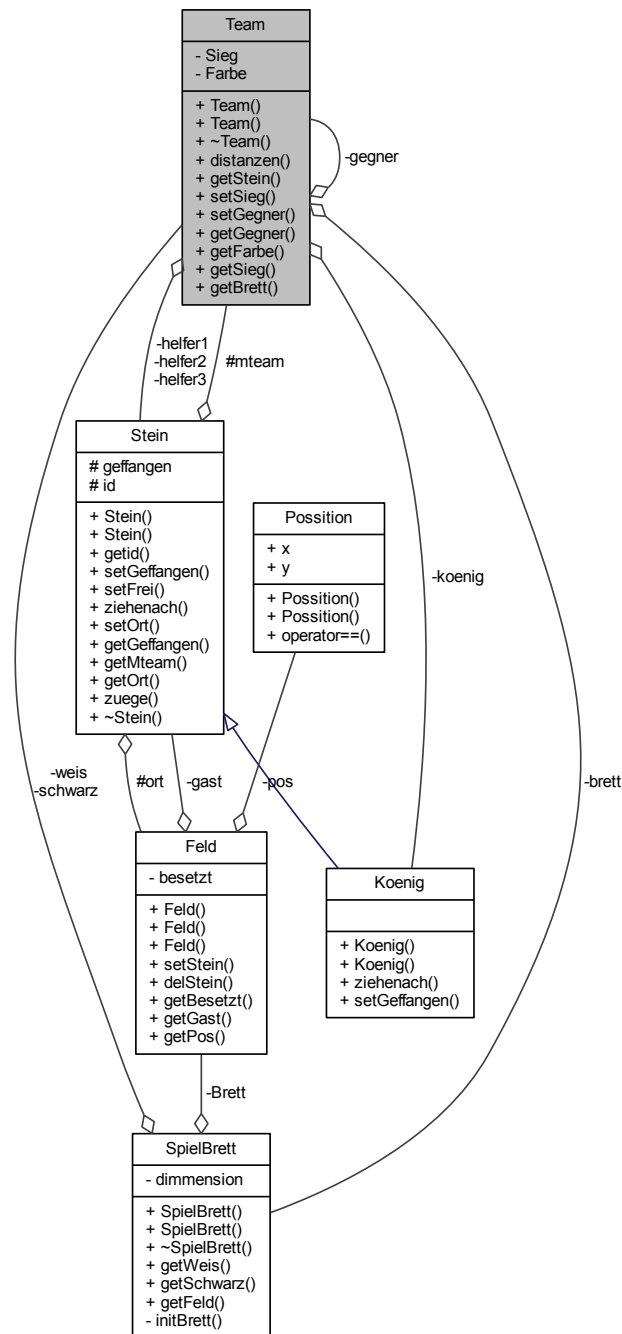
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [Strategie.h](#)
- [Strategie.cpp](#)

4.13 Team Klassenreferenz

```
#include <Team.h>
```

Zusammengehörigkeiten von Team:



Öffentliche Methoden

- `Team (SpielBrett *br, bool f, Feld *s1, Feld *s2, Feld *s3, Feld *k, Team *g)`
- `Team ()=default`
- `virtual ~Team ()`
- `void distanzen (const Stein &anfrage, int *arr)`
- `Stein & getStein (int id) const`

- void `setSieg` (bool new_var)
- void `setGegner` (Team *new_var)
- Team * `getGegner` () const
- bool `getFarbe` () const
- bool `getSieg` ()
- SpielBrett * `getBrett` () const

Private Attribute

- Stein * `helfer1` =nullptr
- Stein * `helfer2` =nullptr
- Stein * `helfer3` =nullptr
- Koenig * `koenig` =nullptr
- bool `Sieg` =false
- Team * `gegner` =nullptr
- SpielBrett * `brett` =nullptr
- bool `Farbe` =false

4.13.1 Ausführliche Beschreibung

class `Team`

4.13.2 Beschreibung der Konstruktoren und Destruktoren

4.13.2.1 `Team::Team (SpielBrett * br, bool f, Feld * s1, Feld * s2, Feld * s3, Feld * k, Team * g = nullptr)`

Erzeugt `Team`.

4.13.2.2 `Team::Team ()` [default]

4.13.2.3 `Team::~~Team ()` [virtual]

4.13.3 Dokumentation der Elementfunktionen

4.13.3.1 `void Team::distanzen (const Stein & anfrage, int * arr)`

`distanzen()` Trägt x und y Distanzen der "Anderen" Steine in einem Array ein. Array muss 6 Felder besitzen und vom Typ Integer sein.

Parameter

in	<i>&anfrage</i>	: <code>Stein</code> , [out] *arr : int array[6]
----	---------------------	--

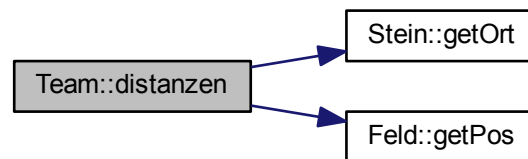
Rückgabe

unsigned short

Parameter

<i>anfrage</i>

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.13.3.2 **SpielBrett * Team::getBrett () const**

4.13.3.3 **bool Team::getFarbe () const**

4.13.3.4 **Team * Team::getGegner () const**

Gibt Pointer auf Gegnerisches [Team](#) aus.

4.13.3.5 **bool Team::getSieg ()**

4.13.3.6 **Stein & Team::getStein (int *id*) const**

getStein Gibt Referenz auf [Stein](#) mit übergebener ID zurück, bei falschen ID's wird Referenz auf [Koenig](#) zurückgegeben. 1-3 -> Helfer 4 -> [Koenig](#)

Parameter

<i>in</i>	<i>id</i>	: int
-----------	-----------	-------

Rückgabe

&[Stein](#)

4.13.3.7 **void Team::setGegner (Team * *new_var*)**

Setze Gegnerisches [Team](#)

4.13.3.8 **void Team::setSieg (bool *new_var*)**

Set the value of Sieg

Parameter

<i>new_var</i>	the new value of Sieg
----------------	-----------------------

4.13.4 Dokumentation der Datenelemente

4.13.4.1 **SpielBrett* Team::brett = nullptr** [*private*]

4.13.4.2 `bool Team::Farbe =false` `[private]`

4.13.4.3 `Team* Team::gegner =nullptr` `[private]`

4.13.4.4 `Stein* Team::helfer1 =nullptr` `[private]`

4.13.4.5 `Stein * Team::helfer2 =nullptr` `[private]`

4.13.4.6 `Stein * Team::helfer3 =nullptr` `[private]`

4.13.4.7 `Koenig* Team::koenig =nullptr` `[private]`

4.13.4.8 `bool Team::Sieg =false` `[private]`

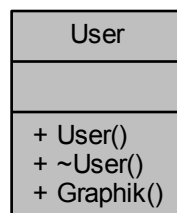
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [Team.h](#)
- [Team.cpp](#)

4.14 User Klassenreferenz

```
#include <User.h>
```

Zusammengehörigkeiten von User:



Öffentliche Methoden

- [User](#) ()
- virtual [~User](#) ()
- void [Graphik](#) ()

4.14.1 Ausführliche Beschreibung

```
class User
```

4.14.2 Beschreibung der Konstruktoren und Destruktoren

4.14.2.1 `User::User ()`

Empty Constructor

4.14.2.2 `User::~~User ()` [virtual]

Empty Destructor

4.14.3 Dokumentation der Elementfunktionen

4.14.3.1 `void User::Graphik ()`

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [User.h](#)
- [User.cpp](#)

4.15 zug Strukturreferenz

```
#include <zug.h>
```


Öffentliche Attribute

- `Feld * zu` =nullptr
- `Stein * stein` =nullptr
- `int wert` =100
- `Possition zpos`

4.15.1 Ausführliche Beschreibung

struct Zug Daten Struktur die einen Spiel-Zug Symbolisiert.

4.15.2 Beschreibung der Konstruktoren und Destruktoren

4.15.2.1 `zug::zug()` [default]

4.15.2.2 `zug::zug(Feld * z, Stein * s)` [inline]

4.15.2.3 `zug::zug(const zug & z)` [inline]

4.15.3 Dokumentation der Elementfunktionen

4.15.3.1 `bool zug::operator<(const zug & z) const` [inline]

Kleiner als Operator Vergleicht Zuege nach Wertigkeit;

Parameter

<code>z</code>	
----------------	--

Rückgabe

4.15.3.2 `zug& zug::operator=(const zug & z)` [inline]

4.15.3.3 `bool zug::operator==(const zug & z) const` [inline]

Vergleichs-Operator Vergleicht Zuege auf gleiche Ziel-Position

Parameter

<code>z</code>	
----------------	--

Rückgabe

4.15.4 Dokumentation der Datenelemente

4.15.4.1 `Stein* zug::stein` =nullptr

4.15.4.2 `int zug::wert` =100

4.15.4.3 `Possition zug::zpos`

4.15.4.4 Feld* zug::zu =nullptr

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [zug.h](#)

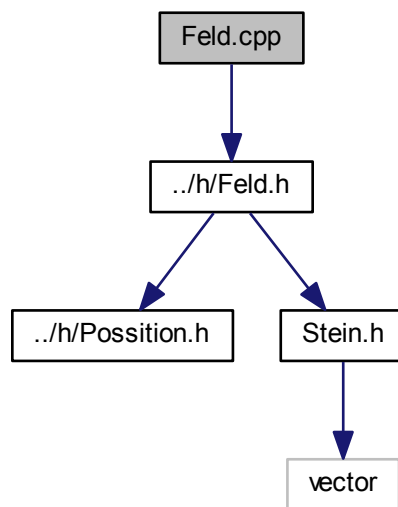
Kapitel 5

Datei-Dokumentation

5.1 Feld.cpp-Dateireferenz

```
#include "../h/Feld.h"
```

Include-Abhängigkeitsdiagramm für Feld.cpp:



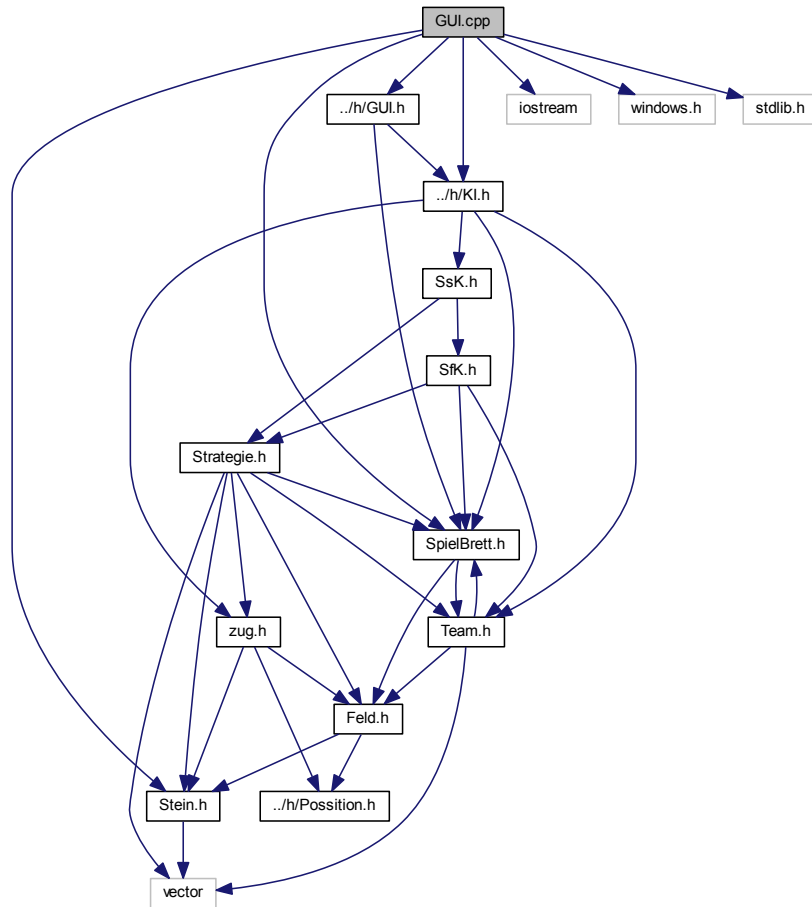
Makrodefinitionen

- `#define` [STEIN_C](#)

5.1.1 Makro-Dokumentation

5.1.1.1 `#define` STEIN_C

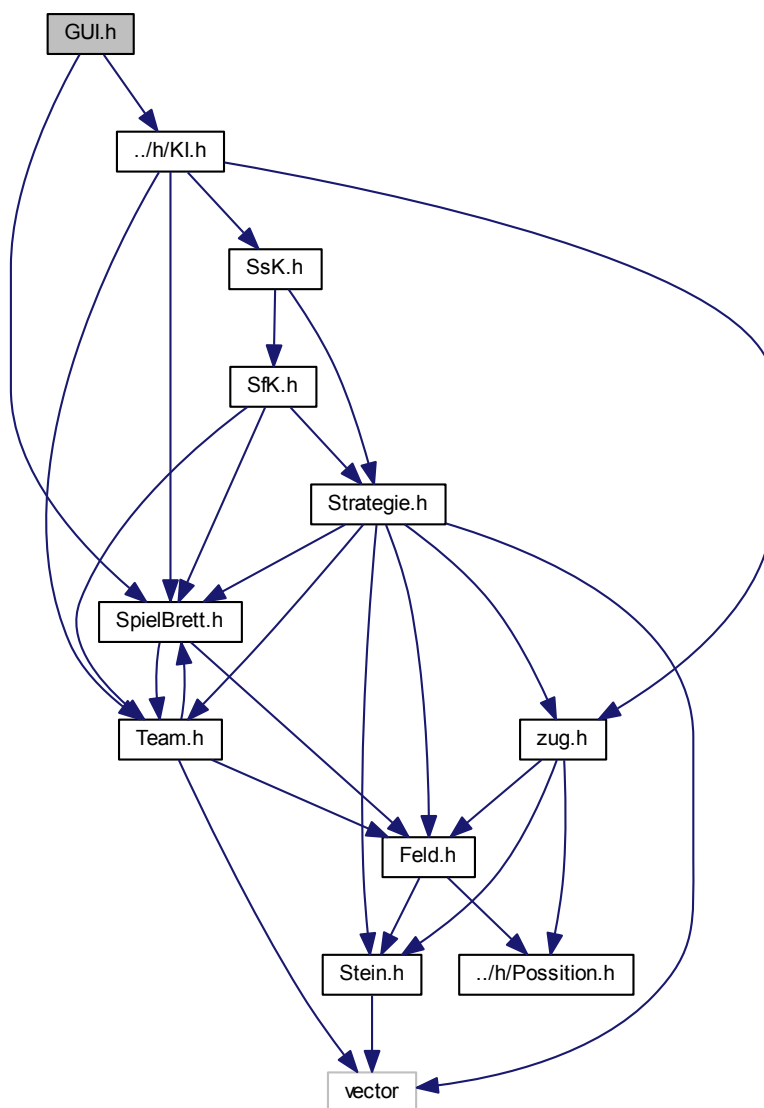

```
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include "../h/KI.h"
#include "../h/Spielbrett.h"
#include "../h/Stein.h"
Include-Abhängigkeitsdiagramm für GUI.cpp:
```



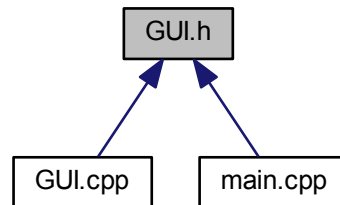
5.4 GUI.h-Dateireferenz

```
#include "SpielBrett.h"
#include "../h/KI.h"
```

Include-Abhängigkeitsdiagramm für GUI.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



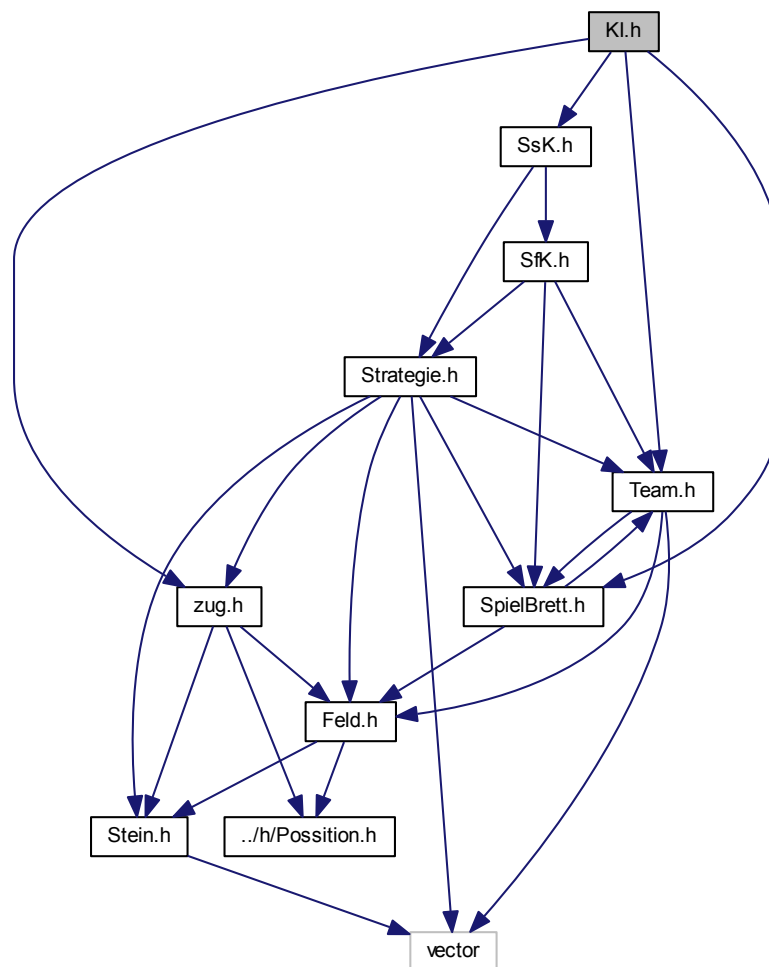
Klassen

- class [GUI](#)

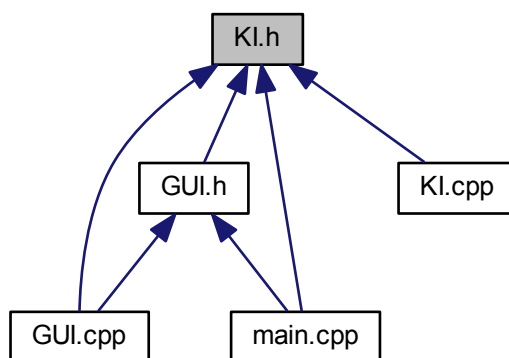
5.5 KI.cpp-Dateireferenz

```
#include "../h/KI.h"
#include "../h/SfK.h"
#include "../h/SsK.h"
#include "../h/SfH.h"
#include "../h/SrH.h"
#include "../h/Strategie.h"
#include <iostream>
#include <cstdlib>
#include <time.h>
#include <algorithm>
```


Include-Abhängigkeitsdiagramm für KI.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



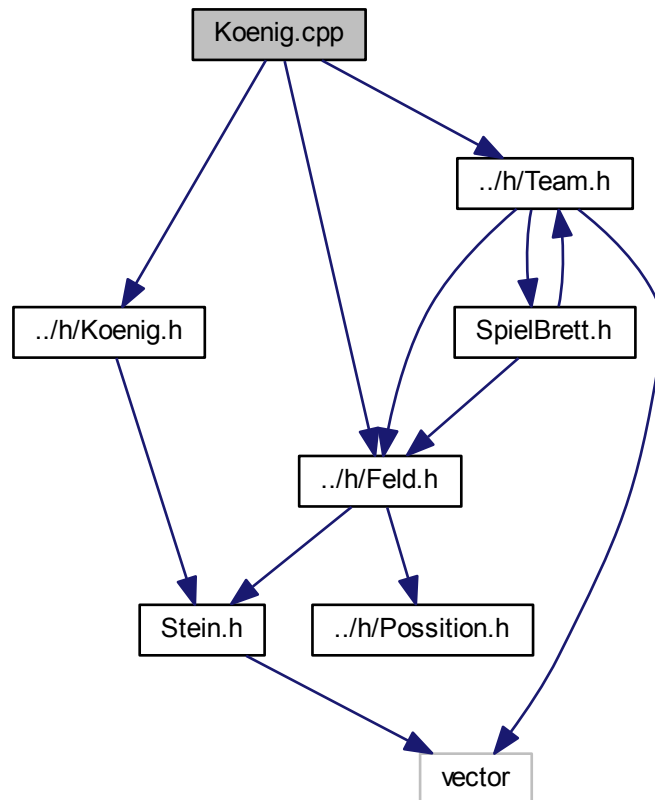
Klassen

- class [KI](#)

5.7 Koenig.cpp-Dateireferenz

```
#include "../h/Koenig.h"  
#include "../h/Feld.h"  
#include "../h/Team.h"
```


Include-Abhängigkeitsdiagramm für Koenig.cpp:



Makrodefinitionen

- #define `KOEING_C`

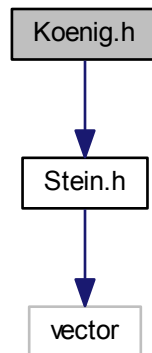
5.7.1 Makro-Dokumentation

5.7.1.1 #define KOEING_C

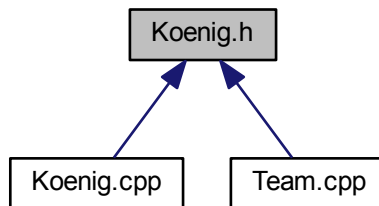
5.8 Koenig.h-Dateireferenz

```
#include "Stein.h"
```

Include-Abhängigkeitsdiagramm für Koenig.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class [Koenig](#)

Makrodefinitionen

- #define [KOENIG_H](#)

5.8.1 Makro-Dokumentation

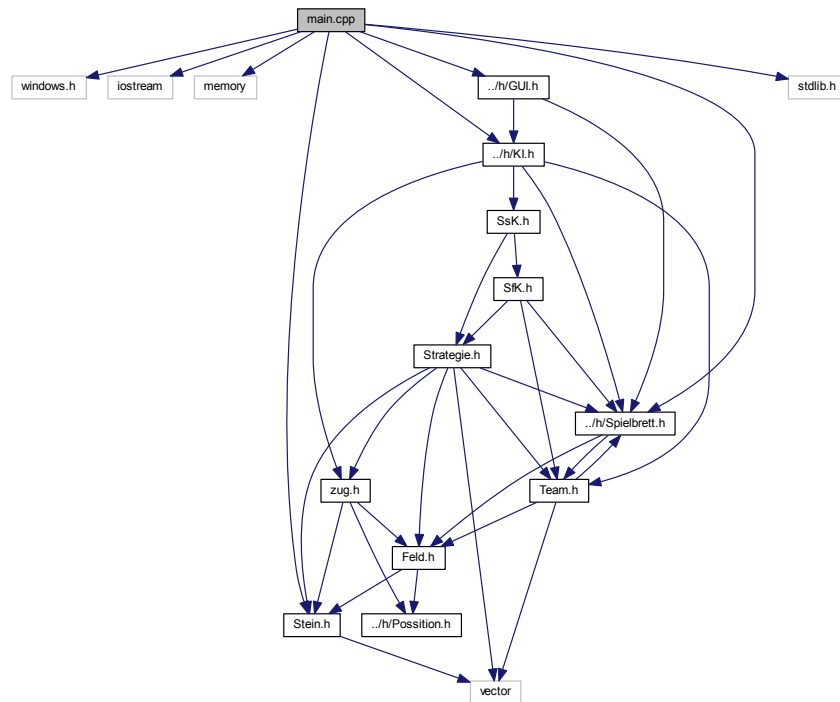
5.8.1.1 #define KOENIG_H

5.9 main.cpp-Dateireferenz

```
#include <windows.h>
```

```
#include <iostream>
#include <memory>
#include "../h/Spielbrett.h"
#include <stdlib.h>
#include "../h/KI.h"
#include "../h/GUI.h"
#include "../h/Stein.h"
```

Include-Abhängigkeitsdiagramm für main.cpp:



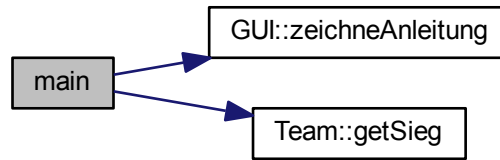
Funktionen

- int **main** (int _argc, char *argv[])

5.9.1 Dokumentation der Funktionen

5.9.1.1 `int main (int _argc, char * argv[])`

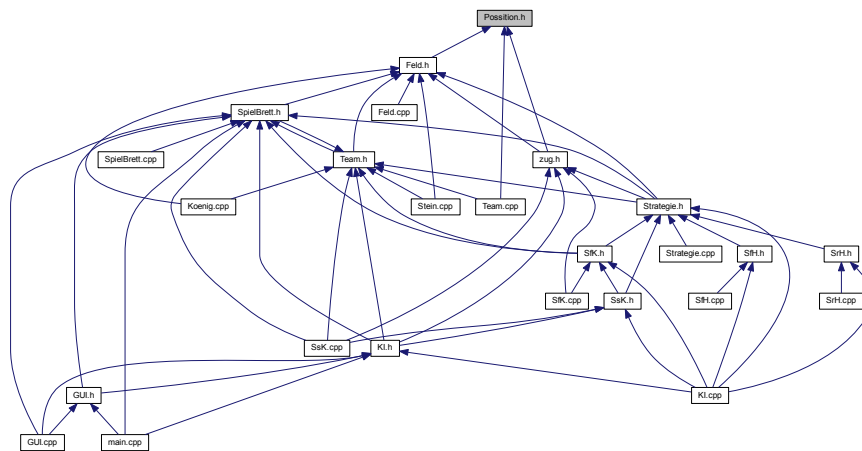
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



5.10 Main.h-Dateireferenz

5.11 Possition.h-Dateireferenz

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- struct [Position](#)

Makrodefinitionen

- `#define` [POSSITION_H](#)

5.11.1 Makro-Dokumentation

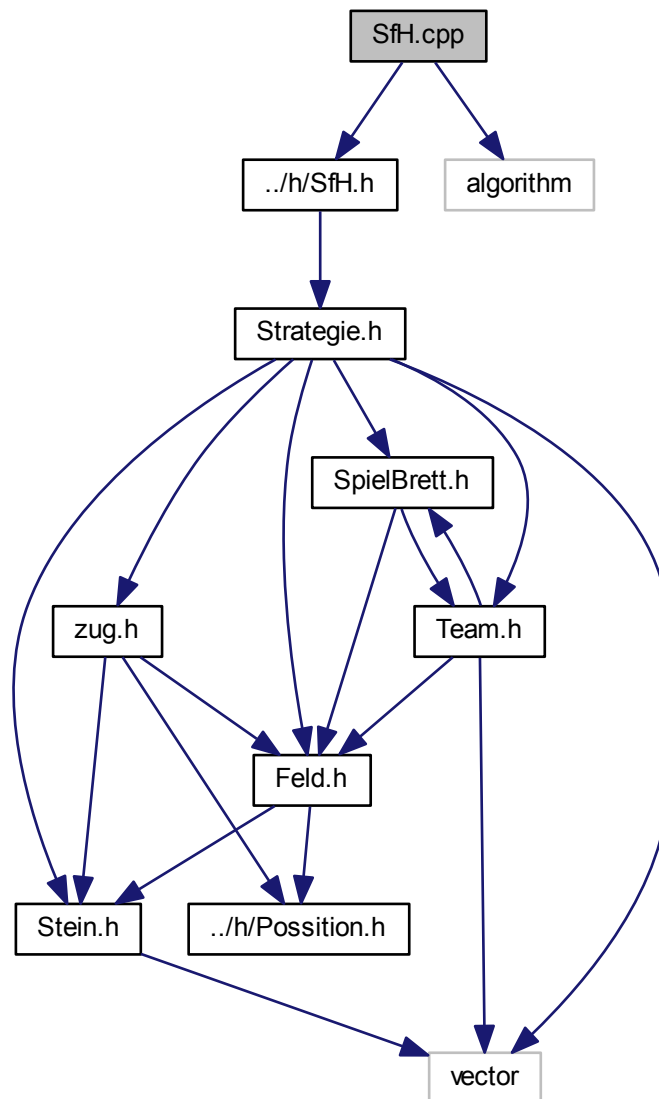
5.11.1.1 `#define POSSITION_H`

5.12 SfH.cpp-Dateireferenz

```
#include "../h/SfH.h"
```

```
#include <algorithm>
```

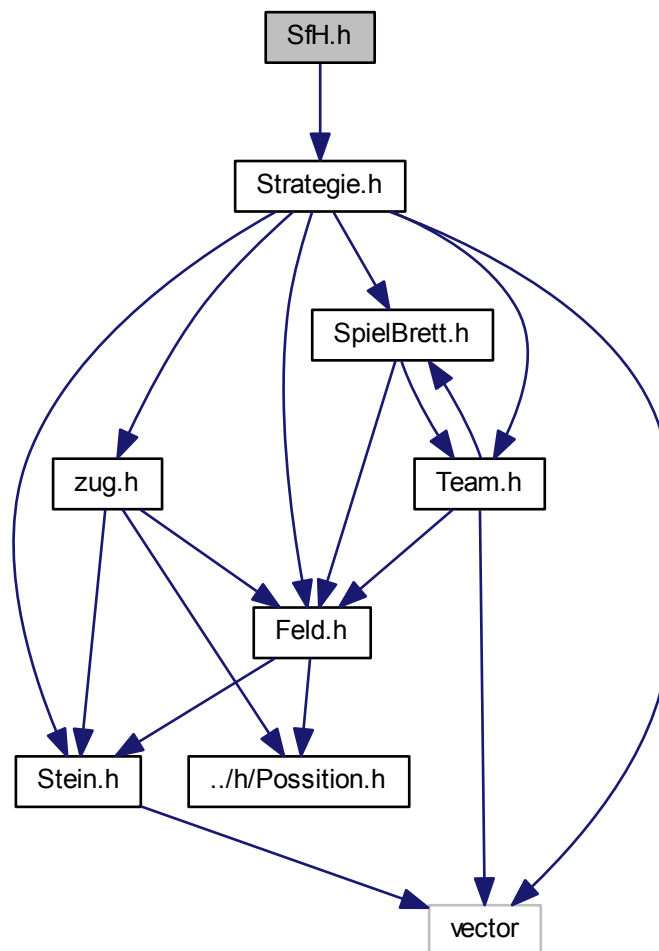
Include-Abhängigkeitsdiagramm für SfH.cpp:



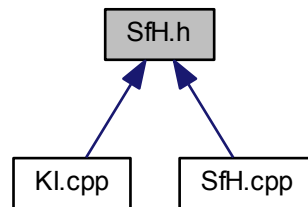
5.13 SfH.h-Dateireferenz

```
#include "Strategie.h"
```

Include-Abhängigkeitsdiagramm für SfH.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



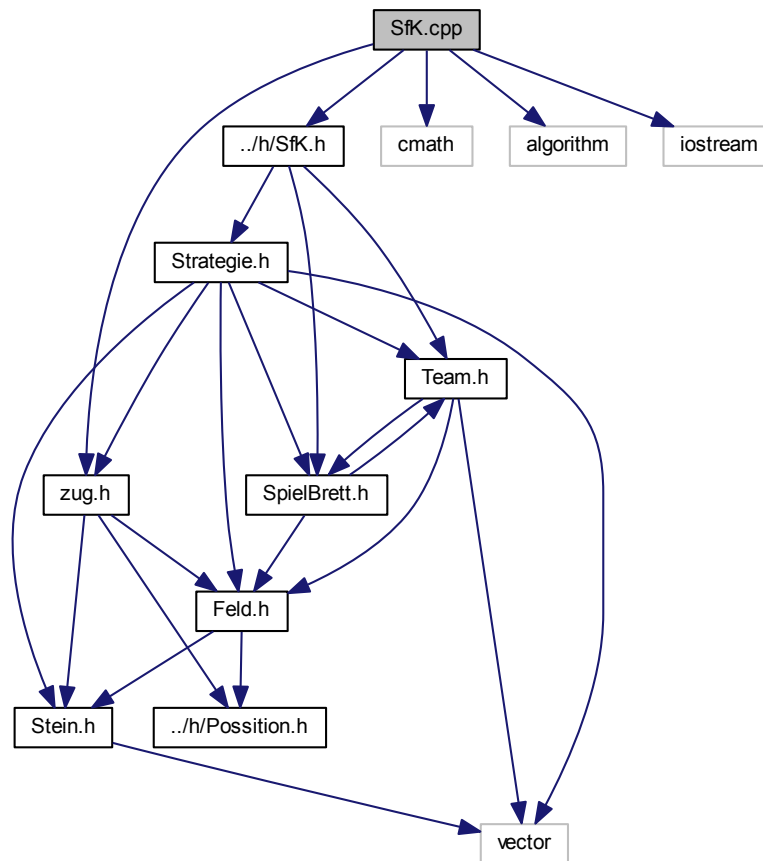
Klassen

- class [SfH](#)

5.14 SfK.cpp-Dateireferenz

```
#include "../h/SfK.h"  
#include <cmath>  
#include <algorithm>  
#include "../h/zug.h"  
#include <iostream>
```

Include-Abhängigkeitsdiagramm für SfK.cpp:



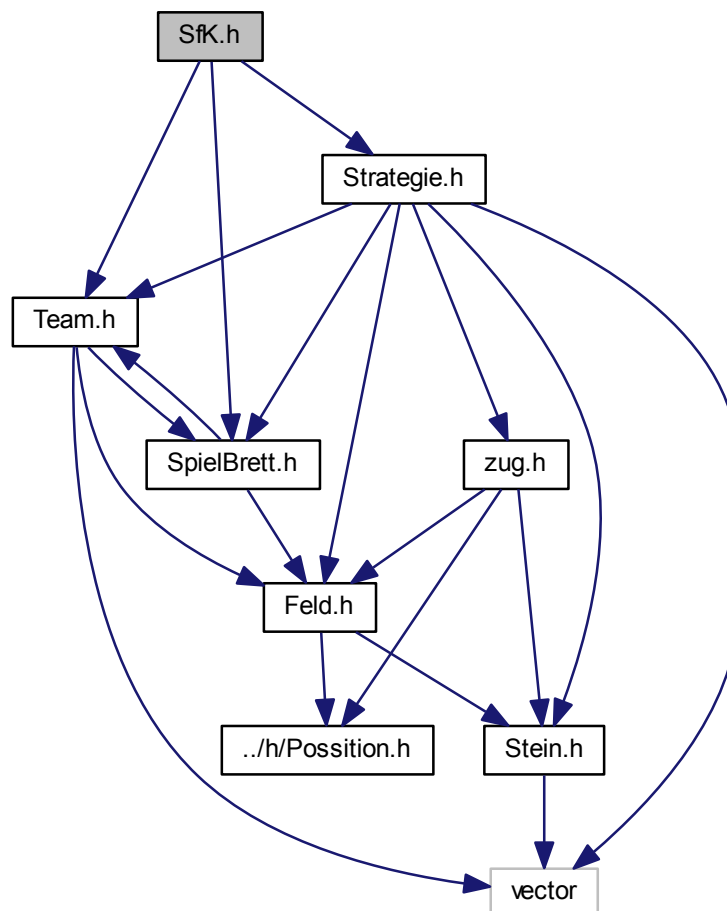
5.15 SfK.h-Dateireferenz

```

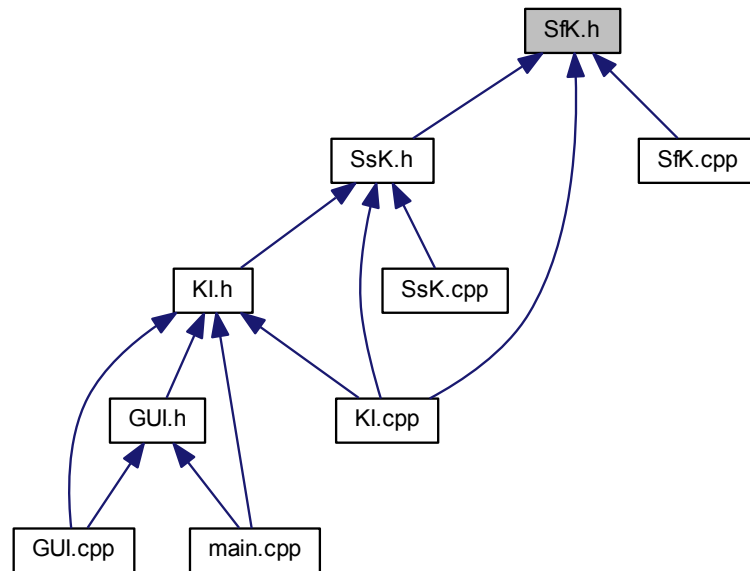
#include "Team.h"
#include "SpielBrett.h"
#include "Strategie.h"

```


Include-Abhängigkeitsdiagramm für SfK.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



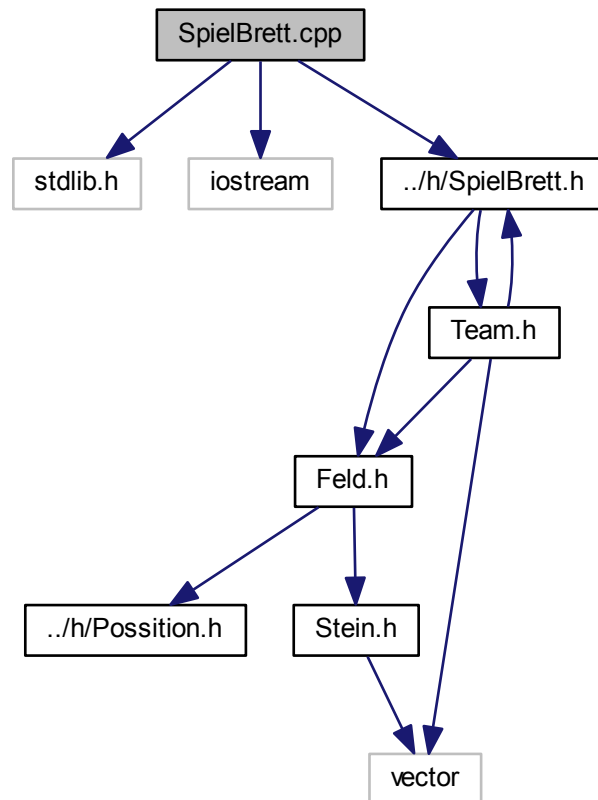
Klassen

- class [SfK](#)

5.16 SpielBrett.cpp-Dateireferenz

```
#include <stdlib.h>
#include <iostream>
#include "../h/SpielBrett.h"
```

Include-Abhängigkeitsdiagramm für SpielBrett.cpp:



Makrodefinitionen

- #define SPIELBRETT_C

5.16.1 Makro-Dokumentation

5.16.1.1 #define SPIELBRETT_C

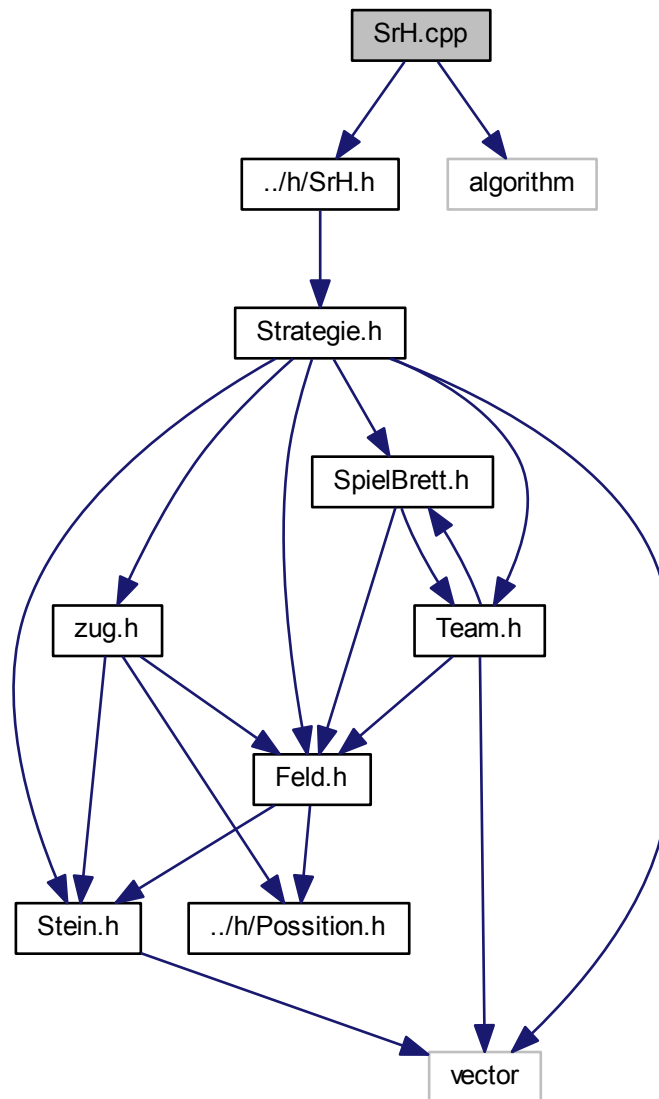
5.17 SpielBrett.h-Dateireferenz

```
#include "Feld.h"
#include "Team.h"
```



```
#include <algorithm>
```

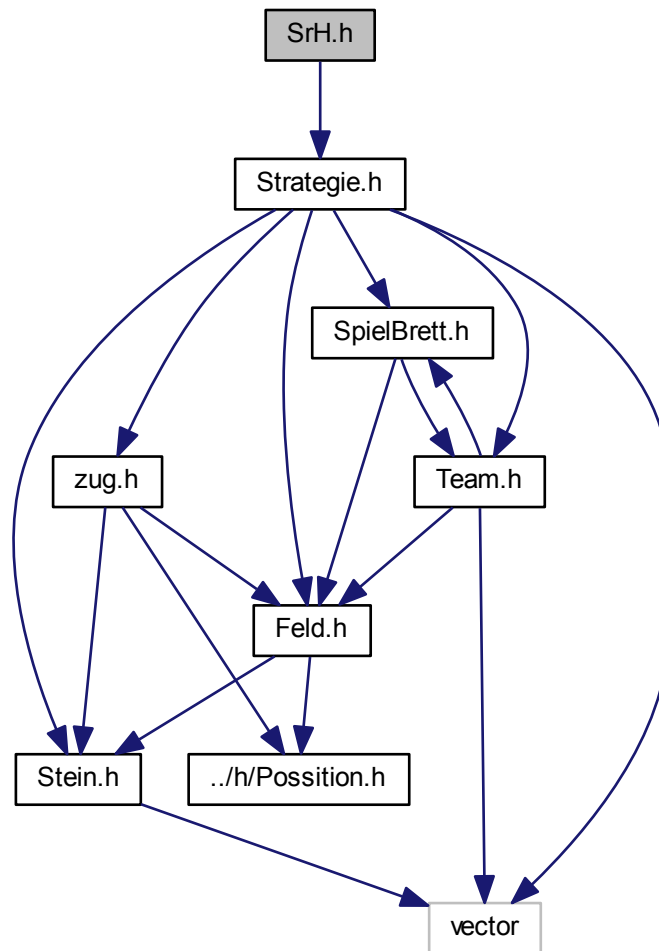
Include-Abhängigkeitsdiagramm für SrH.cpp:



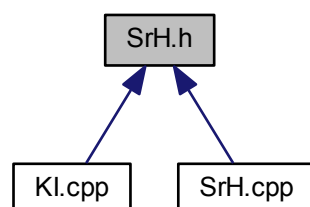
5.19 SrH.h-Dateireferenz

```
#include "Strategie.h"
```

Include-Abhängigkeitsdiagramm für SrH.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



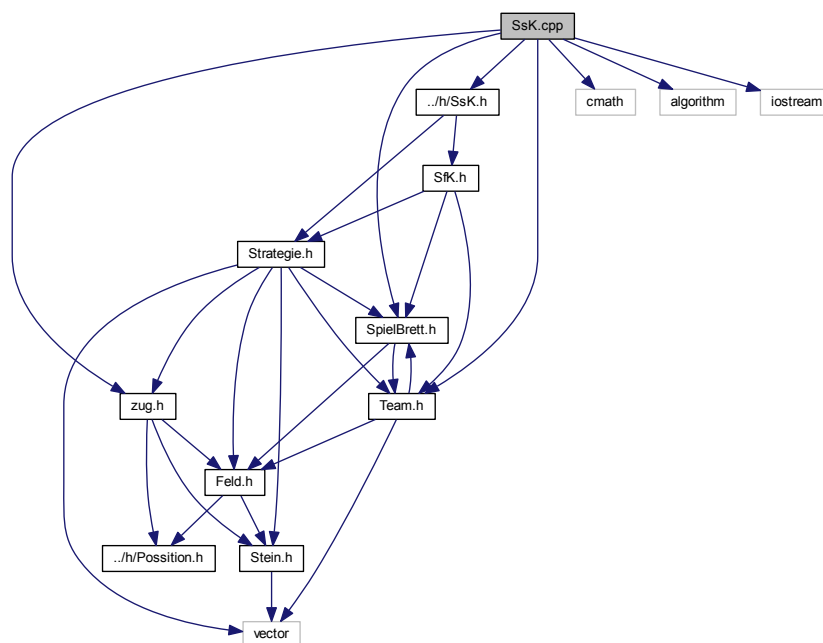
Klassen

- class [SrH](#)

5.20 SsK.cpp-Dateireferenz

```
#include "../h/SsK.h"
#include <cmath>
#include <algorithm>
#include "../h/zug.h"
#include "../h/SpielBrett.h"
#include "../h/Team.h"
#include <iostream>
```

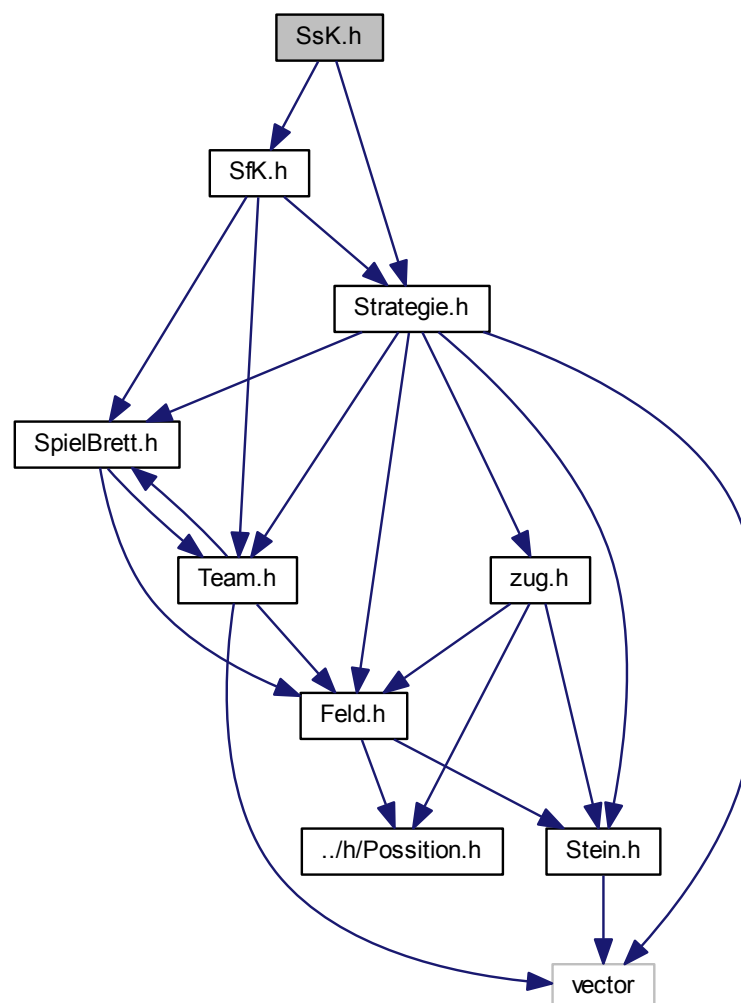
Include-Abhängigkeitsdiagramm für SsK.cpp:



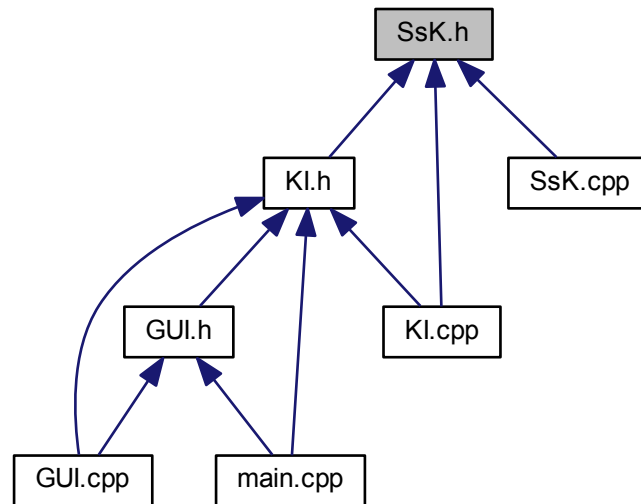
5.21 SsK.h-Dateireferenz

```
#include "Strategie.h"
#include "SfK.h"
```

Include-Abhängigkeitsdiagramm für SsK.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



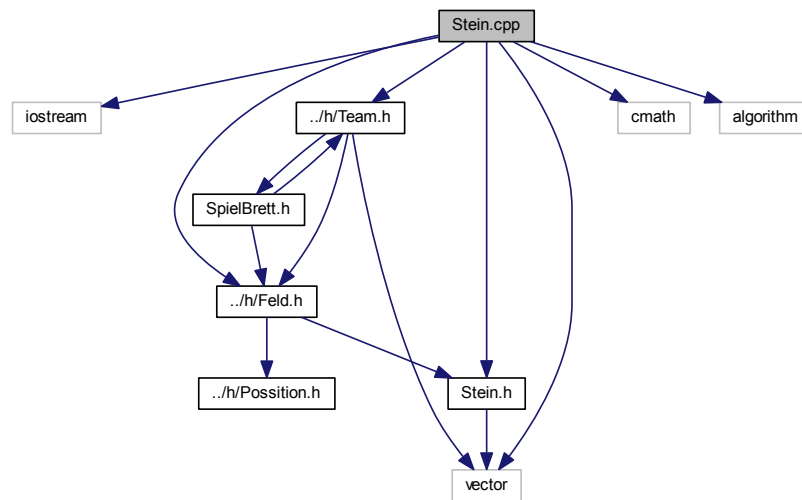
Klassen

- class [SsK](#)

5.22 Stein.cpp-Dateireferenz

```
#include <iostream>
#include "../h/Feld.h"
#include "../h/Team.h"
#include "../h/Stein.h"
#include <cmath>
#include <vector>
#include <algorithm>
```

Include-Abhängigkeitsdiagramm für Stein.cpp:



Makrodefinitionen

- #define [STEIN_C](#)

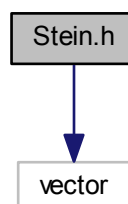
5.22.1 Makro-Dokumentation

5.22.1.1 #define STEIN_C

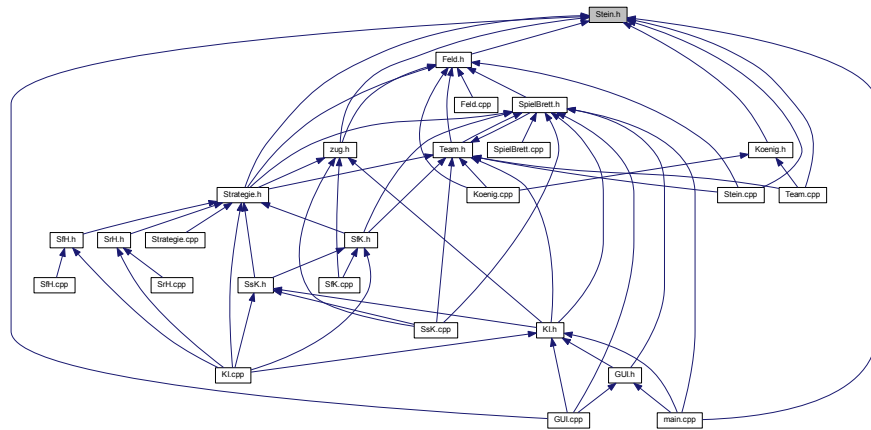
5.23 Stein.h-Dateireferenz

```
#include <vector>
```

Include-Abhängigkeitsdiagramm für Stein.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



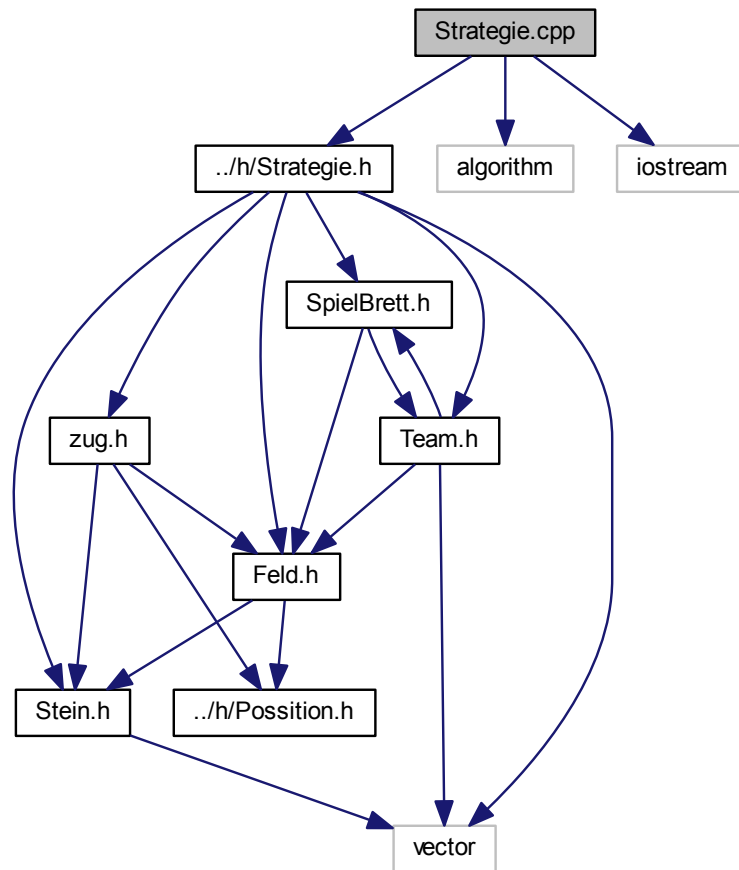
Klassen

- class [Stein](#)

5.24 Strategie.cpp-Dateireferenz

```
#include "../h/Strategie.h"
#include <algorithm>
#include <iostream>
```

Include-Abhängigkeitsdiagramm für Strategie.cpp:



5.25 Strategie.h-Dateireferenz

```
#include <vector>
#include "Feld.h"
#include "SpielBrett.h"
#include "Team.h"
#include "Stein.h"
#include "zug.h"
```

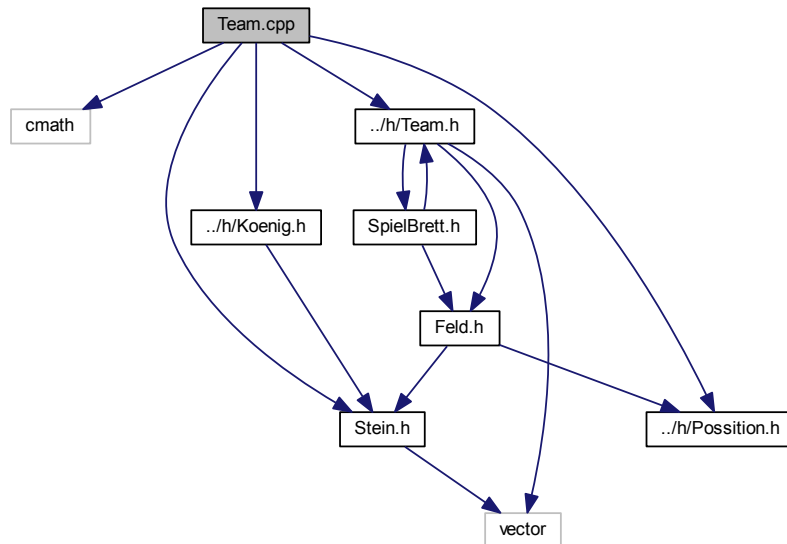

Klassen

- class [Strategie](#)

5.26 Team.cpp-Dateireferenz

```
#include <cmath>
#include "../h/Team.h"
#include "../h/Stein.h"
#include "../h/Koenig.h"
#include "../h/Possition.h"
```

Include-Abhängigkeitsdiagramm für Team.cpp:



Makrodefinitionen

- #define [TEAM_C](#)

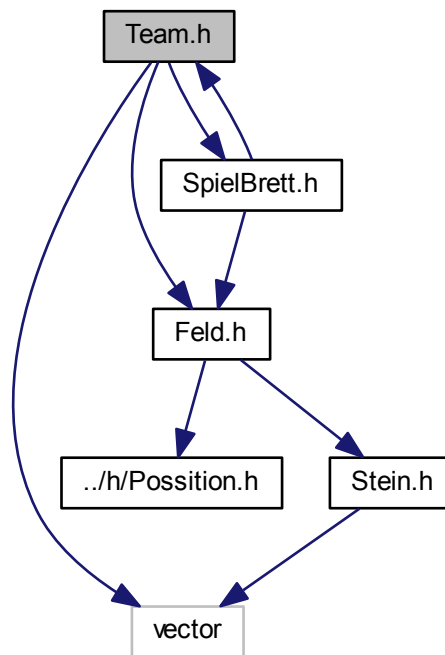
5.26.1 Makro-Dokumentation

5.26.1.1 #define TEAM_C

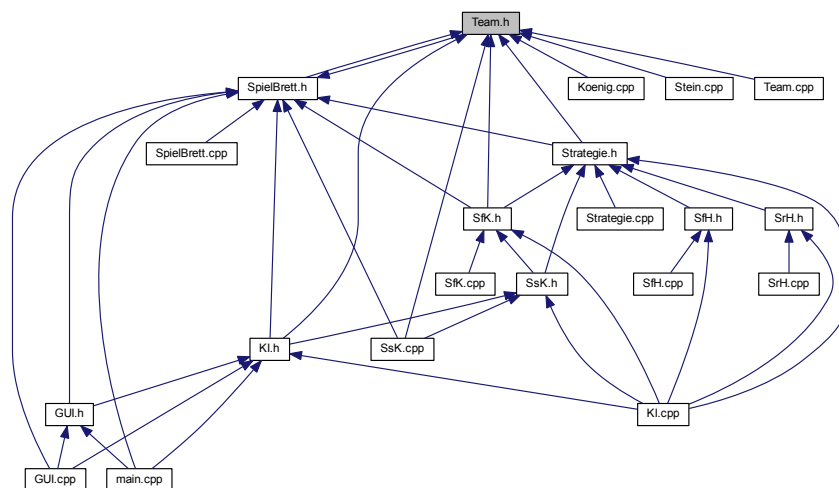
5.27 Team.h-Dateireferenz

```
#include <vector>
#include "Feld.h"
#include "SpielBrett.h"
```

Include-Abhängigkeitsdiagramm für Team.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



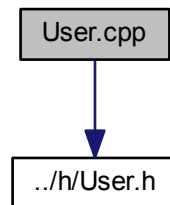
Klassen

- class [Team](#)

5.28 User.cpp-Dateireferenz

```
#include "../h/User.h"
```

Include-Abhängigkeitsdiagramm für User.cpp:



Makrodefinitionen

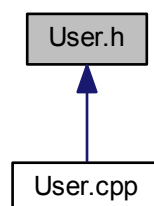
- #define `USER_C`

5.28.1 Makro-Dokumentation

5.28.1.1 #define USER_C

5.29 User.h-Dateireferenz

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



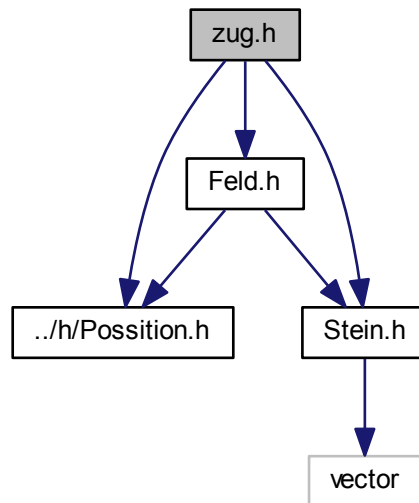
Klassen

- class `User`

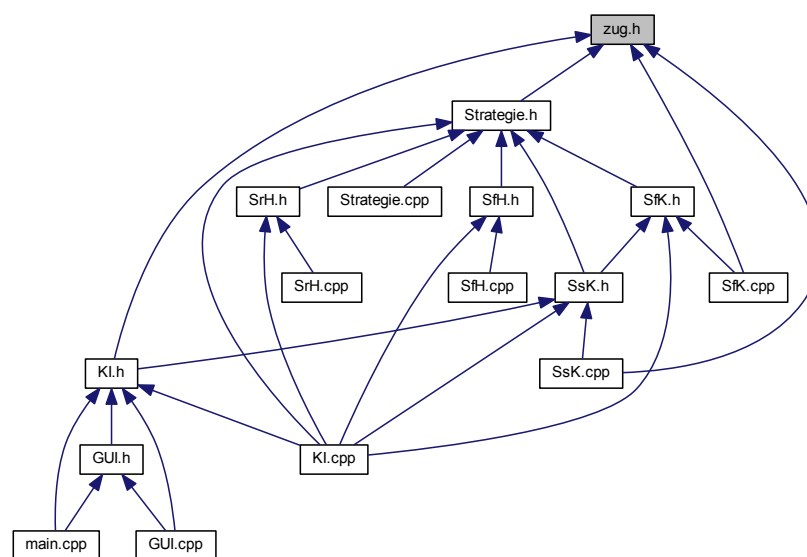
5.30 zug.h-Dateireferenz

```
#include "Feld.h"
#include "Stein.h"
#include "Possition.h"
```

Include-Abhängigkeitsdiagramm für zug.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- struct [zug](#)

Index

- ~KI
 - KI, [15](#)
- ~SfH
 - SfH, [27](#)
- ~SfK
 - SfK, [31](#)
- ~SpielBrett
 - SpielBrett, [35](#)
- ~SrH
 - SrH, [39](#)
- ~SsK
 - SsK, [42](#)
- ~Stein
 - Stein, [46](#)
- ~Strategie
 - Strategie, [51](#)
- ~Team
 - Team, [55](#)
- ~User
 - User, [58](#)
- aZuege
 - Strategie, [52](#)
- abrett
 - KI, [19](#)
- anzstrat
 - KI, [19](#)
- besetzt
 - Feld, [10](#)
- bewerten
 - SfH, [27](#)
 - SfK, [31](#)
 - SrH, [39](#)
 - SsK, [42](#)
 - Strategie, [51](#)
- Brett
 - SpielBrett, [36](#)
- brett
 - GUI, [12](#)
 - Strategie, [52](#)
 - Team, [56](#)
- delStein
 - Feld, [9](#)
- dimension
 - SpielBrett, [36](#)
- distanzen
 - Team, [55](#)
- dv
 - KI, [19](#)
- Farbe
 - Team, [56](#)
- Feld, [7](#)
 - besetzt, [10](#)
 - delStein, [9](#)
 - Feld, [9](#)
 - gast, [10](#)
 - getBesetzt, [9](#)
 - getGast, [9](#)
 - getPos, [9](#)
 - pos, [10](#)
 - setStein, [10](#)
- Feld.cpp, [63](#)
 - STEIN_C, [63](#)
- Feld.h, [64](#)
- GUI, [10](#)
 - brett, [12](#)
 - GUI, [12](#)
 - Klsw, [12](#)
 - Spieler, [12](#)
 - zeichneAnleitung, [12](#)
 - zeichneSpielfeld, [12](#)
 - zeichneZug, [12](#)
- GUI.cpp, [64](#)
- GUI.h, [65](#)
- gZuege
 - SsK, [43](#)
- gast
 - Feld, [10](#)
- geffangen
 - Stein, [48](#)
- gegner
 - SsK, [43](#)
 - Team, [57](#)
- getBesetzt
 - Feld, [9](#)
- getBrett
 - KI, [15](#)
 - Team, [56](#)
- getFarbe
 - Team, [56](#)
- getFeld
 - SpielBrett, [35](#)
- getGast
 - Feld, [9](#)
- getGeffangen
 - Stein, [47](#)

- getGegner
 - Team, 56
- getMteam
 - Stein, 47
- getOrt
 - Stein, 47
- getPos
 - Feld, 9
- getSchwarz
 - SpielBrett, 35
- getSieg
 - Team, 56
- getStein
 - Team, 56
- getTeam
 - KI, 15
- getWeis
 - SpielBrett, 35
- getWert
 - Strategie, 52
- getZuege
 - Strategie, 52
- getid
 - Stein, 47
- getmZuege
 - Strategie, 51
- Graphik
 - User, 58
- h1
 - Strategie, 52
- h2
 - Strategie, 52
- h3
 - Strategie, 52
- helfer1
 - Team, 57
- helfer2
 - Team, 57
- helfer3
 - Team, 57
- id
 - Stein, 48
- initBrett
 - SpielBrett, 36
- k
 - Strategie, 52
- KI, 13
 - ~KI, 15
 - abrett, 19
 - anzstrat, 19
 - dv, 19
 - getBrett, 15
 - getTeam, 15
 - KI, 15
 - mergeStrategie, 15, 16
 - nZug, 19
 - nexZug, 17
 - seachBestZug, 18
 - strat, 19
 - t, 19
 - KI.cpp, 67
 - KI.h, 68
 - Klsw
 - GUI, 12
 - KOEING_C
 - Koenig.cpp, 71
 - KOENIG_H
 - Koenig.h, 72
 - Koenig, 20
 - Koenig, 22
 - setGeffangen, 22
 - ziehenach, 22
 - koenig
 - Team, 57
 - Koenig.cpp, 70
 - KOEING_C, 71
 - Koenig.h, 71
 - KOENIG_H, 72
- mZuege
 - Strategie, 52
- main
 - main.cpp, 73
- main.cpp, 72
 - main, 73
- Main.h, 74
- mergeStrategie
 - KI, 15, 16
- mteam
 - Stein, 48
- nZug
 - KI, 19
 - Strategie, 52
- nexZug
 - KI, 17
 - Strategie, 52
- operator<
 - zug, 60
- operator=
 - zug, 60
- operator==
 - Possition, 24
 - zug, 60
- ort
 - Stein, 49
- POSSITION_H
 - Possition.h, 74
- pos
 - Feld, 10
- posSicher
 - SsK, 42
- Possition, 23

- operator==, 24
- Possition, 24
- x, 24
- y, 24
- Possition.h, 74
- POSSITION_H, 74
- SPIELBRETT_C
 - SpielBrett.cpp, 81
- STEIN_C
 - Feld.cpp, 63
 - Stein.cpp, 88
- schwarz
 - SpielBrett, 36
- seachBestZug
 - KI, 18
- setFrei
 - Stein, 47
- setGeffangen
 - Koenig, 22
 - Stein, 47
- setGegner
 - Team, 56
- setOrt
 - Stein, 47
- setSieg
 - Team, 56
- setStein
 - Feld, 10
- SfH, 24
 - ~SfH, 27
 - bewerten, 27
 - SfH, 27
- SfH.cpp, 75
- SfH.h, 76
- SfK, 28
 - ~SfK, 31
 - bewerten, 31
 - SfK, 31
- SfK.cpp, 77
- SfK.h, 78
- Sieg
 - Team, 57
- SpielBrett, 33
 - ~SpielBrett, 35
 - Brett, 36
 - dimension, 36
 - getFeld, 35
 - getSchwarz, 35
 - getWeis, 35
 - initBrett, 36
 - schwarz, 36
 - SpielBrett, 34
 - weis, 36
- SpielBrett.cpp, 80
 - SPIELBRETT_C, 81
- SpielBrett.h, 81
- Spieler
 - GUI, 12
- SrH, 36
 - ~SrH, 39
 - bewerten, 39
 - SrH, 39
- SrH.cpp, 82
- SrH.h, 83
- SsK, 40
 - ~SsK, 42
 - bewerten, 42
 - gZuege, 43
 - gegner, 43
 - posSicher, 42
 - SsK, 42
- SsK.cpp, 85
- SsK.h, 85
- Stein, 43
 - ~Stein, 46
 - geffangen, 48
 - getGeffangen, 47
 - getMteam, 47
 - getOrt, 47
 - getid, 47
 - id, 48
 - mteam, 48
 - ort, 49
 - setFrei, 47
 - setGeffangen, 47
 - setOrt, 47
 - Stein, 46
 - ziehenach, 47
 - zuege, 48
- stein
 - zug, 60
- Stein.cpp, 87
 - STEIN_C, 88
- Stein.h, 88
- strat
 - KI, 19
- Strategie, 49
 - ~Strategie, 51
 - aZuege, 52
 - bewerten, 51
 - brett, 52
 - getWert, 52
 - getZuege, 52
 - getmZuege, 51
 - h1, 52
 - h2, 52
 - h3, 52
 - k, 52
 - mZuege, 52
 - nZug, 52
 - nexZug, 52
 - Strategie, 51
 - team, 52
 - wert, 52
- Strategie.cpp, 89
- Strategie.h, 90

t

KI, 19

TEAM_C

Team.cpp, 92

Team, 53

~Team, 55

brett, 56

distanzen, 55

Farbe, 56

gegner, 57

getBrett, 56

getFarbe, 56

getGegner, 56

getSieg, 56

getStein, 56

helfer1, 57

helfer2, 57

helfer3, 57

koenig, 57

setGegner, 56

setSieg, 56

Sieg, 57

Team, 55

team

Strategie, 52

Team.cpp, 92

TEAM_C, 92

Team.h, 92

USER_C

User.cpp, 94

User, 57

~User, 58

Graphik, 58

User, 57

User.cpp, 94

USER_C, 94

User.h, 94

weis

SpielBrett, 36

wert

Strategie, 52

zug, 60

x

Position, 24

y

Position, 24

zeichneAnleitung

GUI, 12

zeichneSpielfeld

GUI, 12

zeichneZug

GUI, 12

ziehenach

Koenig, 22

Stein, 47

zpos

zug, 60

zu

zug, 60

zuege

Stein, 48

zug, 58

operator<, 60

operator=, 60

operator==, 60

stein, 60

wert, 60

zpos, 60

zu, 60

zug, 60

zug.h, 95