

„Sudoku Solver“

Programmierprojekt aus Praktikum aus Programmierung

1. Problembeschreibung

1.1 Problemdefinition

Ziel des Projekts ist es ein Programm zu schreiben das ein Sudoku-Rätsel lösen kann. Die Eingabe soll durch ein graphisches Interface erfolgen. Dabei muss die Angabe übertragen werden und das Rätsel soll durch einen Algorithmus iterativ gelöst werden. Ist die Angabe nicht ausreichend bestimmt, müssen eventuell noch weitere Eingaben vorgenommen werden.

1.2 Erklärung Sudoku

Bei dem Rätsel handelt es sich um eine Matrix aus 9 x 9 Felder bei der es gilt in

- jeder Reihe
- jeder Spalte
- jeder Sub-Matrix (9 Bereiche zu je 3 x 3 Feldern)

jede Ziffer von 1 bis 9 einzusetzen.

	1	2	3	4	5	6	7	8	9
1									
2		1			2			3	
3									
4									
5		4			5			6	
6									
7									
8		7			8			9	
9									

Abbildung 1: Schema Sudoku

2. Analyse und Modellierung

2.1 Beschreibung der Problemlösung

Um ein Sudoku zu lösen müssen zuerst die vorhandenen Daten in ein zweidimensionales Array eingegeben werden. In dem Array soll zu jedem Feld gespeichert ob es schon gelöst ist, wie viele mögliche Lösungen es noch gibt und falls es nur noch eine Lösung gibt, soll der Wert extra abgespeichert werden.

Wird einem Feld ein endgültiger Wert zugewiesen, werden in allen Feldern derselben Reihe, derselben Spalte sowie desselben Sub-Feldes, die restlichen möglichen Werte reduziert.

Durch Durchgehen jedes Feldes des gesamten Feldes wird in mehreren Iterationen geschaut ob andere Felder einen definitiven Wert zugewiesen bekommen und so die möglichen Lösungen weiter reduziert werden können.

2.2 Use Case

Use-Case Name: Lösen eines Sudokus

Akteur: Rätselliebhaber

Vorbedingungen: Rätsel und Stift vorhanden

Nachbedingungen: Keine

Auslöser: Wille und Lust zum Lösen des Rätsels

Ablaufbeschreibung:

- Zeitung öffnen
- Rätsel anschauen
- Felder durchgehen und nach möglichen Lösungen suchen
- Durch lösen einzelner Felder werden neue Lösungsmöglichkeiten sichtbar
- Rätsel fertig gelöst
- Nachkontrolle ob alle Zeilen, Spalten und Sub-Felder richtig gelöst wurden

Fehlsituationen:

- Rätsel ist zu schwer und kann nicht gelöst werden
- Fehleingabe bei Lösung
- Rätsel hat keine eindeutig bestimmte Lösung
- Zeit für Rätsel reicht nicht

Variationen:

- Rätsel wird mit Zeitunterbrüchen gelöst
- Nachkontrolle findet nicht statt

Instanz:

Florian hat Langeweile und möchte wieder einmal ein Rätsel lösen. Er schlägt die Zeitung auf und siehe da – er findet ein Sudoku. Er sucht sich einen Kugelschreiber aus seiner Schreibtischlade heraus und schaut sich das Rätsel einmal an. Er erkennt, dass im mittleren Bereich des Sudokus viele Angaben vorhanden sind und kann ein zusätzliches Feld eindeutig bestimmen. Dies erlaubt ihm wiederum ein weiteres Feld eindeutig zu bestimmen, ...

Nachdem alle Felder gelöst sind, schaut Florian sich noch mal ein ob alle Zeilen, Spalten und Sub-Felder wirklich richtig gelöst wurden und das die Gesamtlösung stimmt.

Ergebnisse:

- Lösung des Rätsels

Nicht funktionale Anforderungen:

- Zufriedenheit des Rätselfuchs

Autor: Florian Marschoun, Eigenbeschreibung am 5. Dezember 2005

2.3 Klassendiagramm

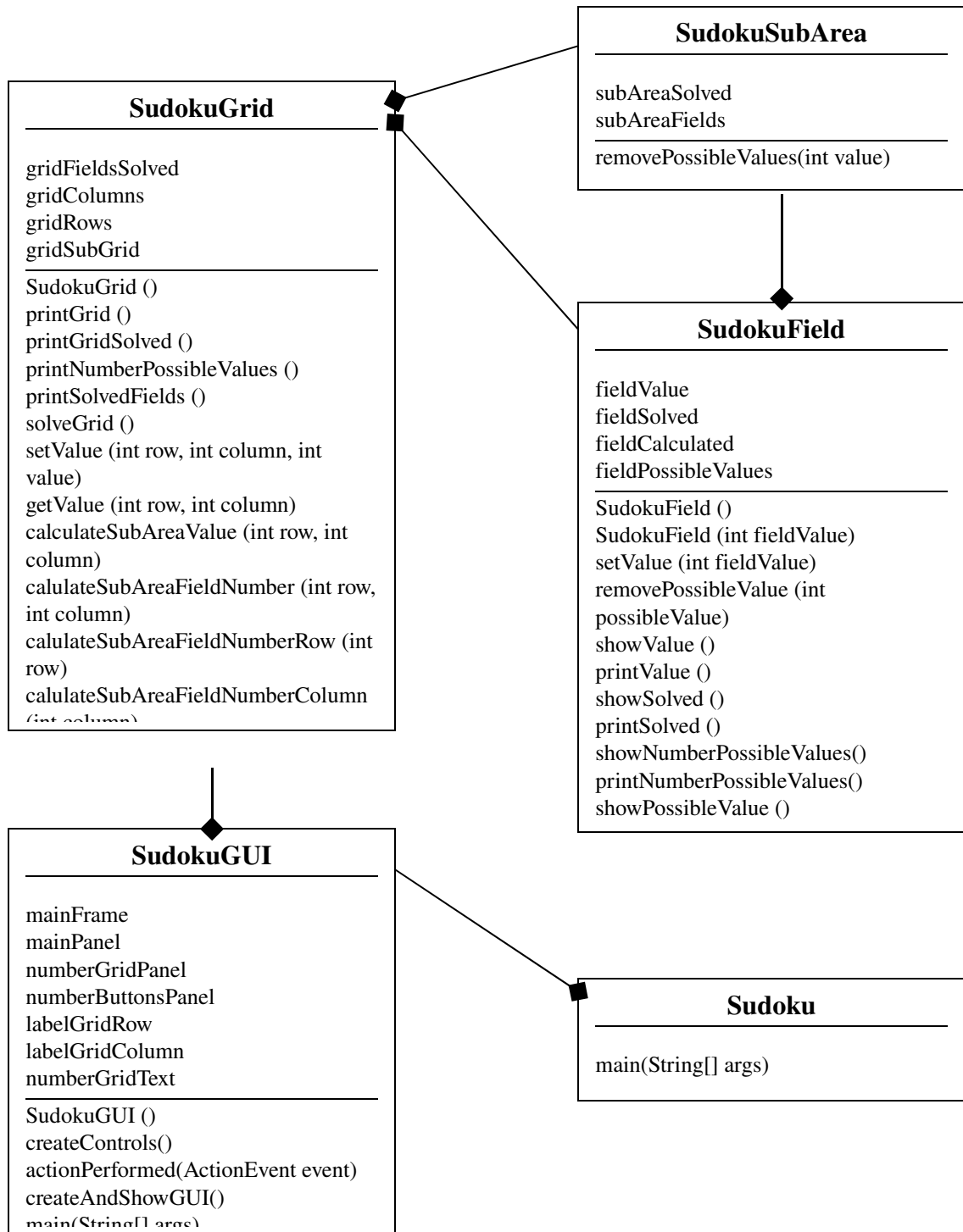
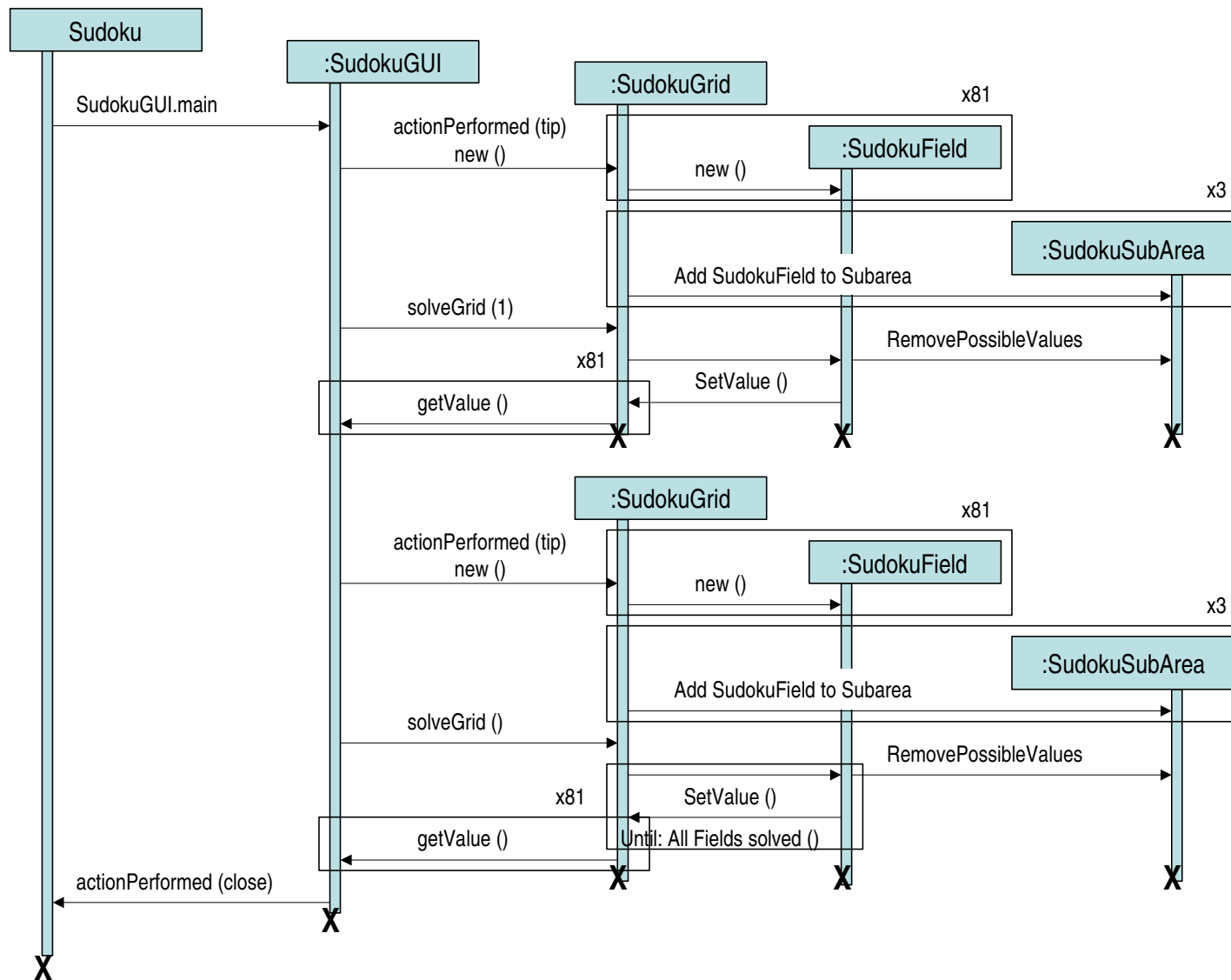


Abbildung 2: Klassendiagramm

2.4 Sequenzdiagramm (Abbildung 3):



2.5 Beschreibung der Klassen:

Sudoku: Hauptklasse deren Main-Methode beim Starten des Programms ausgeführt wird und das GUI initialisiert.

SudokuGUI: Diese Klasse beinhaltet die Benutzeroberfläche des Sudoku Solvers. Es werden sowohl die graphischen Element definiert sowie die Verbindung zur Subklasse SudokuGrid hergestellt, die die eigentlichen Lösungsalgorithmen beinhaltet.

SudokuGrid: Diese Matrix baut aus den Klasse SudokuField und SudokuSubArea mehrere Matrizen auf, anhand der das Rätsel gelöst werden kann. Dazu wird zuerst eine 9x9 Feld-Matrix erstellt und dann die einzelnen Felder jeweils einer SubArea für die Zeile, Spalte und Untergruppe (3x3 Matrix im 9x9 Feld) zugewiesen.

In der Folge stellt diese Klasse die Methoden zur Lösung des Rätsels zur Verfügung.

SudokuField: Diese Klasse beinhaltet die Daten für die einzelnen Felder, unter anderem ob eine Feld schon als gelöst gilt, wie viele weiter Möglichkeiten es noch gibt, usw. Sie wird durch die Klasse SudokuGrid initialisiert.

SudokuSubArea: Dieser Klasse werden die einzelnen Felder (SudokuField) zugeordnet um die Lösungsmöglichkeiten für einzelne Sub-Gruppen auszunützen (Zeile, Spalte, Untergruppe).

3. Benützung des Programms

Nach dem Starten des Programms wird das folgende Fenster sichtbar:

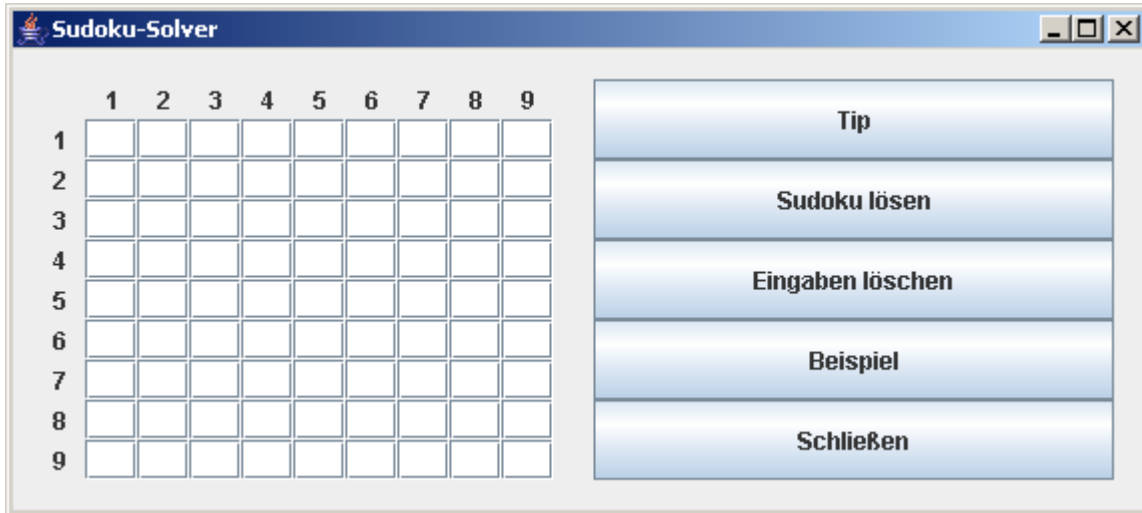


Abbildung 3: GUI des Solvers

Im linken Bereich findet sich die Eingabemaske für die Zahlen-Werte des Sudokus. Bereits bekannte Zahlenwerte müssen in das jeweilige Feld eingetragen werden. Unbekannte Felder müssen leer gelassen werden. Auf der rechten Seite sind die Bedienelemente in folgender Reihe übereinander angeordnet:

- **Tip:** Ist eine Angabe in die Eingabe-Maske übernommen kann hier eine Iteration des Lösung Algorithmus ausgeführt werden und die Matrix des Rätsels wird um einige Werte ergänzt.
- **Sudoku lösen:** Sofern es möglich ist wird das gesamte Rätsel gelöst.
- **Eingabe löschen:** Dieser Button löscht alle eingegebenen Werte aus der Matrix.
- **Beispiel:** Eintragen eines Beispiel-Sudokus in die Matrix.
- **Schließen:** Beenden des Programms.