Technische Universität Berlin **WS 2013/2014**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–1, Frau Klink

Dr. Sebastian Stiller
Dr. Max Klimm
Jan-Philipp Kappmeier
Georg Loho

Katharina Bütow, Christian Döblin, Alexander Hopp, Judith Kubitza, Daniel Kuske, Antje Lehmann, Steffen Przybylowicz, Olivia Röhrig, Robert Rudow, Daniel Schmand, Hendrik Schrezenmaier, Judith Simon, Sebastian Spies, Jan Zur

Computerorientierte Mathematik I

7. Programmieraufgabe

Abnahme: spätestens am 23./24. 01. 2014 (je nach Gruppennummer).

Gerade Gruppennummern geben spätestens am Donnerstag, ungerade Gruppennummern geben spätestens am Freitag ab.

Sudokus sind Zahlenrätsel, bei denen ein (in der Regel ein quadratisches 9×9-)Gitter vorgegeben ist, in dem einige Kästchen mit Zahlen von $1, 2, \ldots, 9$ ausgefüllt sind. Die Aufgabe besteht darin, in die restlichen Kästchen Zahlen $1, 2, \ldots, 9$ einzutragen, so dass

- in jeder Zeile jede Zahl genau einmal vorkommt,
- in jeder Spalte jede Zahl genau einmal vorkommt und
- in jedem (durch die dicken Linien abgegrenzten) 3×3-Quadrat jede Zahl genau einmal vorkommt.

In dieser Programmieraufgabe schreiben wir ein Programm, das Sudoku-Rätsel völlig automatisch lösen kann. Leere Kästchen stellen wir dabei mit der Ziffer 0 dar.

(a) Schreibt eine Klasse `Sudoku`, welche ein 9×9-Sudoku modelliert; auf der Homepage gibt es bereits eine Vorlage dazu. Die Klasse soll folgenden Konstruktor und folgende Methoden besitzen:

- eine Klassenmethode

```
public static boolean isValid(int[][] grid)
```

die überprüft, ob das übergebene Array ein *ausgefülltes* Sudoku darstellt. Dazu muss überprüft werden, ob das Array die Dimension 9×9 hat, ob Zahlen < 1 oder > 9 im Array stehen und ob Zahlen in einer Zeile / in einer Spalte / in einem Block *genau einmal* vorkommen.

Hinweis: Diese Methode ist in der Vorgabe bereits vorhanden!

- eine Methode

  `public boolean isGridValid()`

  die überprüft, ob ein Sudoku gültig ist, das heißt ob das Array die Dimension 9×9 hat, ob Zahlen < 0 oder > 9 im Array stehen und ob Zahlen in einer Zeile / in einer Spalte / in einem Block doppelt vorkommen.

  Hinweis: Diese Methode stimmt fast mit der darüber überein, nur dass leere Zellen erlaubt sind.

- einen Konstruktor

  `public Sudoku(int[][] grid)`

  welcher ein neues Sudoku basierend auf dem übergebenen zweidimensionalen Array erzeugt. Dabei soll überprüft werden, ob der Array Inkonsistenzen enthält (z. B. wenn Zahlen in einer Zeile / in einer Spalte / in einem Block doppelt vorkommen oder das Array nicht die Dimension 9×9 hat). Werft eine `IllegalArgumentException`, wenn dies der Fall ist.

- eine Methode

  `public int getValue( int row, int column )`

  welche einen Eintrag des Gitters zurückgibt.

- eine Methode

  `public String toString()`

  welche das Sudoku in einer für Menschen lesbaren Textform zurückgibt.

(b) Auf der CoMa-Homepage findet ihr die Sudoku-Sammlung `17_Hints.sudoku` von Gordon Royle von der *University of Western Australia*. Alle Sudokus in dieser Sammlung sind lösbar. Jede Zeile in dieser Datei enthält ein Sudoku-Rätsel, die Ziffern der Zeile entsprechen dabei den anfänglichen Eintragungen im Gitter (von links nach rechts und von oben nach unten). Dabei steht eine 0 für ein leeres Kästchen. Die Zeile

`000000010400000000020000000000050407008000300001090000300400200050100000000806000`

entspricht also dem Sudoku-Rätsel

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

Die Vorgabe enthält die Klasse `SudokuReader`, die eine solche Datei einlesen kann und eine Liste von `Sudoku` zurückgibt. Mit der Klasse aus (a) sollte der Reader kompilieren.

(c) Schreibt eine Klasse `SudokuSolver`, welche den Algorithmus zum Lösen eines Sudokus beinhalten wird. Diese Klasse soll den Konstruktor

```
public SudokuSolver(Sudoku sudoku)
```

besitzen, der das `Sudoku` festlegt, welche die von ihm erzeugte Instanz der Klasse lösen soll. Das Lösen des Sudokus soll von einer Methode

```
public int[][] solve()
```

durchgeführt werden, welche eine Lösung zurückgibt, sofern das `Sudoku` eine Lösung hat, und `null` anderenfalls.

Wie ihr das Sudoku löst, ist euch überlassen. Die einzigen Bedingungen an das von euch verwendete Verfahren sind, dass es prinzipiell jedes Sudoku lösen können muss und dass es schnell genug ist, um die ersten 10 Sudokus aus der vorgegebenen Sammlung in unter 100 Sekunden zu lösen.

Eine einfache Möglichkeit, Sudokus zu Lösen besteht im sogenannten Backtracking. Dabei wird in ein leeres Feld eine der möglichen Ziffern, die eingetragen werden können, testweise gesetzt und der Algorithmus zum Lösen erneut auf der nun um ein zusätzliches ausgefüllte Feld erweiterten Instanz aufgerufen. Wenn so eine Lösung gefunden wird, wird diese ausgegeben. Falls es keine Lösung gibt, muss man die testweise eingetragene Ziffer wieder löschen und durch eine weitere der möglichen Ziffern ersetzen. Wenn keine der möglichen Ziffern zu einer gültigen Lösung führt, muss das leere Feld wiederhergestellt werden und an die Ausführung wird auf der vorherigen Rekursionsstufe weitergeführt.

Schneller ist es, sich für jedes Feld zu merken, welche Ziffern in diesem noch stehen können. Durch die vorgegebenen Ziffern lassen sich in den meisten Feldern einige Ziffern ausschließen. Findet man dabei Felder, in denen nur eine Ziffer möglich ist, kann man diese eintragen und mit ihr in anderen Feldern weitere Ziffern ausschließen. Man kann beide Ansätze auch mischen und nur, wenn kein Feld mit nur einer Möglichkeit mehr existiert einen Backtracking-Schritt machen.

(d) Auf der Homepage findet ihr die Java-Klasse `Main`, die ihr als Blackbox benutzen könnt, um euren Algorithmus zu testen. Diese Klasse erlaubt euch das Laden einer Sudoku-Sammlung und das anschließende Lösen dieser Sudokus mit einem vorgegebenen Zeitlimit.

Hinweis: Ihr könnt euch zur Fehlersuche für euer Verfahren auch eigene Beispiele ausdenken oder aus dem Internet besorgen, wenn bei euch die Sudokus aus der Sammlung zu viel Zeit benötigen. Zum Testen der Klasse aus (a) könnt ihr auch ein voll ausgefülltes Sudoku einlesen lassen.

Viel Spaß und Erfolg!