Praktikumsaufgabe „Sudoku-Generator“

Ein SUDOKU ist ein 9 × 9 – Gitter (s.u., Ziffern als ‚*‘), das mit den Ziffern 1 bis 9 so ausgefüllt ist, dass jede Ziffer

- in jeder Spalte,
- in jeder Zeile und
- in jedem 3 x 3 - Unterblock

genau ein einziges Mal vorkommt.

Insgesamt sind fast 6.7 Trilliarden (10 hoch 21) unterschiedliche SUDOKUS konstruierbar (genau 6.670.903.752.021.072.936.960).

Die Programmieraufgabe bestehe darin, unter Nutzung eines Zufallszahlengenerators SUDOKUS zu erzeugen und anzuzeigen.

Der Generator benutzt den Zufallszahlengenerator zunächst, um ein beliebiges freies Feld zu ermitteln und eine Zahl für dieses Feld auszuwürfeln. Diese Zahl darf natürlich die Spielregeln nicht verletzen, falls doch, nimmt man einfach die nächste.

Durch anschließendes Ausblenden einiger Felder entsteht aus einem SUDOKU ein SUDOKU-Rätsel. Dabei müssen allerdings mindestens 17 Felder unausgeblendet bleiben, um eine eindeutige Lösung zu erhalten.

```
  0 1 2  3 4 5  6 7 8  9 10 11 12
    abc    def    ghi
     0 1 2  3 4 5  6 7 8
   +---+---+---+---+
 1 | * * * | * * * | * * * | 1
 2 | * * * | * * * | * * * | 2
 3 | * * * | * * * | * * * | 3
   +---+---+---+---+         4
 4 | * * * | * * * | * * * | 5
 5 | * * * | * * * | * * * | 6
 6 | * * * | * * * | * * * | 7
   +---+---+---+---+         8
 7 | * * * | * * * | * * * | 9
 8 | * * * | * * * | * * * | 10
 9 | * * * | * * * | * * * | 11
   +---+---+---+---+         12
```

Bsp.:

```
    abc    def    ghi
   +---+---+---+
 1 |123|456|789|
 2 |456|789|123|
 3 |789|123|456|
   +---+---+---+
 4 |231|564|897|
 5 |564|897|231|
 6 |897|231|564|
   +---+---+---+
 7 |312|645|978|
 8 |645|978|312|
 9 |978|312|645|
   +---+---+---+
```

↳ mehrdimensionales Array für Werte 9×9:

```
    0 1 2 3 4 5 6 7 8
  +-------------------+
0 |                   |
1 |                   |
2 |                   |
3 |                   |
4 |                   |
5 |                   |
6 |                   |
7 |                   |
8 |                   |
  +-------------------+
```

```
    abc    def    ghi
   +---+---+---+
 1 |  2|4 6|  8|
 2 |4 6|   |1 3|
 3 |  8|1  |  5|
   +---+---+---+
 4 |2 1|   |8  |
 5 |   |8 7|  3|
 6 |8  |  3|   4|
   +---+---+---+
 7 |  1|   5|   |
 8 |   5|  7|3 2|
 9 |  7|3  |  4|
   +---+---+---+
```

Wir lösen diese Aufgabe in mehreren Teilschritten. Bitte laden Sie für jeden dieser Teilschritte (und nicht mehr) Ihren Lösungsstand in OneDrive hoch. Erzeugen Sie sich im Netbeans für das nächste Projekt jeweils eine Kopie Ihres letzten Schrittes durch:

Klick rechts auf Projekt<X> -> Copy -> Projekt<X+1>.

Schritt 0: Erste 3 Matrizen ausgeben

Geben Sie die ersten 6 Zeilen des Spielfelds sequentiell aus!

Schritt 1: Spielfeld ausgeben

Lassen Sie sich den Namen des Spielers eingeben, und zeigen Sie ihn am Kopf des Spielfelds an.

Geben Sie das oben links gezeigte Sudoku-Feld inklusive der Zeilen- und Spaltenbeschriftungen auf die Konsole aus. Zeichnen Sie dabei NICHT jede Zeile einzeln, sondern implementieren Sie einen geeigneten Algorithmus!

Schritt 2: Felder manuell belegen

Lassen Sie sich eine Koordinate des Spielfelds und eine Ziffer zwischen 1 und 9 vom Bediener eingeben (z.B. e5, 7), und geben Sie das Spielfeld erneut aus, jedoch mit der Ziffer an der Koordinate des Spielfelds, und allen bereits eingegebenen Ziffern.

Schritt 3: Felder automatisch belegen

Spätestens jetzt sollte Ihre Lösung in einzelne Funktionen zerlegt werden: Draw(), GetRandomPosition(), CheckRules(), WaitUser(), … .

Bauen Sie das Spielfeld aus 9 Matrizen mit je 9 Feldern auf, und arbeiten Sie bei der Regelprüfung mit diesen 9 Matrizen.

Ermitteln Sie nun selbst mit dem Zufallszahlengenerator ein (freies) Feld des Spielfelds zwischen 0 und 80. Benutzen Sie den Zufallszahlengenerator erneut, wenn Sie ein bereits belegtes Feld gewürfelt haben.
Prüfen Sie für dieses Feld nacheinander, ob eine der Zahlen zwischen 1…9 alle 3 Sudokuregeln erfüllt. Nur in dem Fall können Sie sie einsetzen.

> ***Problem**: Sie werden immer wieder an den Punkt kommen, dass jede der Ziffern 1-9 gegen die Regeln verstößt – dann ist Ihre bisherige Zahlenmenge zumindest teilweise ungültig, und Sie müssen ein Backtracking anwenden (die Lösung schrittweise „zurückrollen", analog „Undo"). Wie für eine Unstützung des „Undo" benutzen wir auch hierfür einen Stack (Stapelspeicher), dessen Code inkl. Testroutine Ihnen bereitgestellt wird. Sie legen dabei jede gültig ausgefüllte Feldposition auf den Stack („Push()"). Stossen Sie zum ersten Mal auf eine Sackgasse, entnehmen Sie den obersten Wert vom Stack („Pop()"), und löschen den Wert an dieser Position aus ihrem Spielfeld. Bei der nächsten Sackgasse tun Sie das für die 2 letzten Positionen, dann für 3 etc.*
> *Durch dieses Vorgehen kommen Sie theoretisch bis zu einem leeren Spielfeld zurück, um dem Zufall immer neue Chancen zu geben.*

Merken Sie sich die Werte der bereits belegten Felder. Beenden Sie das Spiel automatisch, wenn das letzte Feld belegt wurde.

Stack.c mit ins Projekt includen

Geben Sie in diesem Schritt nicht mehr das Spielfeld neu nach jedem Würfeln aus, sondern erst nach vollständiger Belegung!

Schritt 4: Sudoku erzeugen

Für die Erzeugung des Rätsels blenden Sie jetzt Zahlen in der Anzeige aus (mindestens 17 und max, 36 Zahlen sollten sichtbar sein).

Merken Sie sich dabei sowohl alle Zahlen Ihrer Lösung, als auch, welche ausgeblendeten Felder Sie für den Spieler freigeben.

Schritt 5: Sudoku lösen

Lassen Sie sich jetzt für die einzelnen Felders des Spielbretts nacheinander Zahlenwerte sagen, und geben Sie sie aus. Visualisieren Sie dabei unterschiedlich, welches die vorgegebenen Werte sind, und welche die des Spielers. Schützen Sie die vorgegebenen Werte gegen Veränderung durch den Spieler!

Bieten Sie über ein Menue folgende Möglichkeiten an:

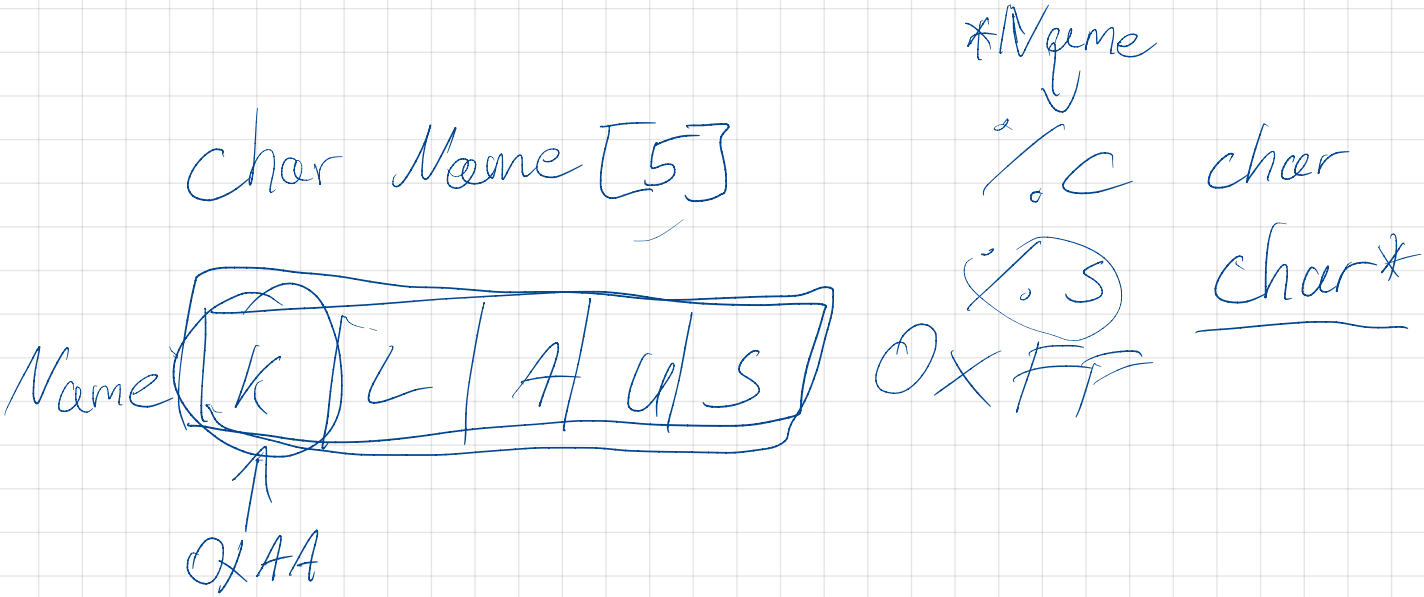
- Letzte Eingabe widerrufen (sollte mehrfach möglich sein, nutzen Sie erneut den Stack)
- Regelprüfung einschalten
- Regelprüfung ausschalten
- Gesamtlösung einblenden
- Gesamtlösung ausblenden
- Spiel beenden

Schritt 6 (optional): Versuch eindeutiger Lösbarkeit

Unsere Vorgehensweise garantiert nicht, dass es nur eine einzige Lösung des Sudokus gibt (es ist also nicht unbedingt eindeutig lösbar)! Optimieren Sie deshalb weiter Ihren Algorithmen zur Erzeugung, z.B. aus dieser Quelle:

https://de.wikipedia.org/wiki/Sudoku#Erstellung_neuer_Sudokus

Viel Spaß und viel Erfolg!


char Name[5]
*Name
Name
K
L
A
d
S
0XAA
0XFF
char
char*