Sudoku

Opdracht Design Patterns 1: 2020/2021

Iedereen heeft ze wel eens gemaakt: een sudoku-puzzel. In een krant of puzzelboekje kom je ze vaak tegen. Ook zijn er al ontelbaar veel applicaties voor geschreven. Jij gaat aan die hoeveelheid nog één applicatie toevoegen, maar nu één die veelzijdig en flexibel is.

In je applicatie kan je helemaal zelf een sudoku oplossen, maar je kan ook geholpen worden door wat hulpgetallen te gebruiken. Daarnaast kan je verschillende soorten Sudoku's spelen: De reguliere 9x9 Sudoku, Sudoku's met andere formaten, Samurai Sudoku en Jigsaw Sudoku.

Opdrachtomschrijving

Je dient een applicatie te maken waarin een Sudoku bronbestand ingelezen wordt die vervolgens weergeven wordt. Je kan er vervolgens voor kiezen om deze Sudoku zelf op te lossen of om de applicatie deze te laten oplossen. Deze functionele eisen zijn te vinden in bijlage 1.

Het belangrijkste doel bij deze opdracht is om onderhoudbare en uitbreidbare code te schrijven. Bij de beoordeling van deze code zullen dus bijvoorbeeld kijken naar de mate van eenvoud waarin nieuwe functionaliteiten gecreëerd zouden kunnen worden. Een aantal voorbeelden hiervan zijn: het maken van een nieuwe web-frontend op de huidige logica, het opslagmedium vervangen door een database, een nieuw type Sudoku kunnen ondersteunen of het ondersteunen van nieuwe oplossingsalgoritmes.
Denk er dus aan nette en modulair opgebouwde code in deze opdracht te gebruiken!

Technische eisen:

- Je applicatie dient in een object-geörienteerde taal te zijn geschreven (C#/Java, elke andere taal dient in overleg te zijn met de docent).
- De onderwezen *design patterns* dienen in je code terug te vinden te zijn (zie *rubric*).
- Er zijn unittests aanwezig om de applicatielogica te testen.
- Je werkt in duo's aan deze applicatie voor een gezamelijk assessment.

Bijlagen

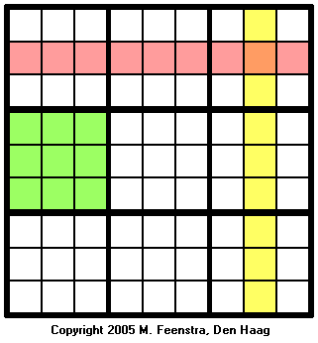
1. Functionele eisen

- *Je moet een Sudokubestand kunnen inlezen.*
  Hierbij moet je de volgende formaten ondersteunen: 4x4, 6x6, 9x9, Samurai Sudoku, en Jigsaw Sudoku
- *Je moet een ingelezen Sudoku kunnen weergeven.*
  Hierbij moet je kunnen kiezen tussen twee soorten weergaven:
  - Eenvoudig: alleen de definitieve cijfers worden getoond.
  - Hulpgetallen: de hulpgetallen die nog niet definitief zijn, worden getoond.
- *Je moet in de applicatie de kunnen wisselen tussen de editorstanden.*
  Editorstanden zijn:
  - Invoeren/Verwijderen van hulpgetallen
  - Invoeren/Verwijderen van definitieve getallen

  In beide van deze standen moet ook gekozen kunnen worden om de applicatie het te laten weergeven als je een foute invoer hebt gegeven.
- *Je moet een cel kunnen selecteren.*
- *Je moet een getal kunnen invoeren of verwijderen in de geselecteerde cel:*
  - In de definitieve-stand tik je een getal in en dan:
    - als de cel leeg was verschijnt dat getal.
    - als dat betreffende getal in de cel stond wordt de cel leeg.
    - als een ander getal in de cel stond dan vervangt dit nieuwe getal het oude getal.
  - In de hulpgetal-stand tik je een getal in en dan:
    - Als de cel een definitief getal bevat gebeurt er niets.
    - Als de cel het betreffende hulpgetal niet reeds bevatte, dan wordt deze toegevoegd aan de cel.
    - Als de cel het betreffende hulpgetal wel reeds bevatte, dan wordt deze verwijderd uit de cel.
- *Je moet kunnen aangeven dat de applicatie (de rest van) de puzzel zelf gaat oplossen.*
  Tip: Gebruik hiervoor het backtracking algoritme dat in de bijlage verder uitgewerkt is.
  Let op: Je applicatie dient open te staan voor andere oplossingsalgoritmes in de toekomst.
- *Je moet kunnen aangeven dat de applicatie de ingevoerde getallen controleert op fouten waarbij hij de incorrect ingevulde getallen direct aangeeft.*

2. Sudokuregels en types

Een Sudoku is een getallenpuzzel in een raster van (typisch) 9 kolommen breed en 9 rijen hoog (Figuur 1). Het raster is verder verdeeld in 9 subrasters van 3 bij 3 velden. In elk veld dient een getal tussen 1 en 9 ingevuld te worden. Hierbij mogen binnen een rij, kolom of subraster geen dubbele getallen voorkomen.
Zie hier voor gedetailleerde spelregels.



Figuur 1 Sudokuregels

4x4 en 6x6 varianten

De varianten met formaten 4x4 en 6x6 werken net zoals de reguliere Sudoku, maar hebben natuurlijk minder velden (Figuren 2 en 3).

	3	4	
4			2
1			3
	2	1	

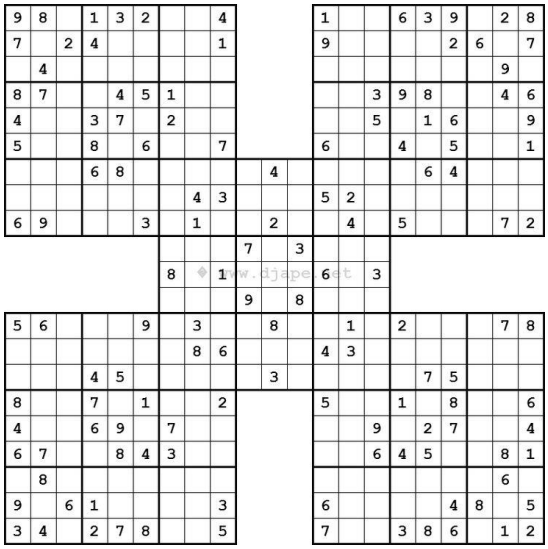
Figuur 3 4x4

		3		1	
5	6		3	2	
	5	4	2		3
2		6	4	5	
	1	2		4	5
	4		1		

Figuur 2 6x6

Samurai Sudoku

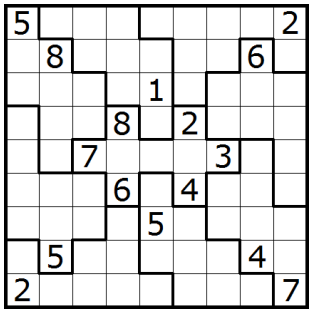
Een Samurai sudoku is een puzzel die bestaat uit 5 overlappende reguliere sudoku puzzels. Het subrooster in de overlappende hoek doet dus mee in beide sudoku puzzels.
Let op: De rijen en kolommen die elkaar verlengen zijn onafhankelijk van elkaar, enkel de 3 velden die ze delen overlappen.



Figuur 4 Samurai

Jigsaw sudoku

Een Jigsaw sudoku is een sudoku van 9 kolommen en 9 rijen waarbij de subroosters niet evenredig verdeeld zijn. In de puzzel is aangegeven welke velden bij welk subrooster horen. Afgezien van de vorm van de subroosters, volgt de Jigsaw sudoku de regels van een reguliere sudoku.



Figuur 5 Jigsaw

3. Bestanden

Voor de sudoku puzzels zijn verschillende bestanden beschikbaar. Ze zullen een extensie hebben die overeenkomt met het type puzzel en een inhoud die hoort bij dat type puzzel.

De mogelijke extensies zijn: *.4x4*, *.9x9*, *.6x6*, *.jigsaw, .samurai.*

De inhoud zal voor een 4x4, 6x6, 9x9 puzzel bestaan uit één rij van getallen. Een 0 betekent dat het een leeg veld betreft, de getallen 1 tot en met 9 zijn vooraf ingevulde waardes. Rijen worden hierin aaneengesloten achter elkaar geplaatst.

(Wil je zelf van zulke bestanden maken, ga dan naar deze website en genereer de gewenste puzzel. Kopieer vervolgens de tekst en vervang in een editor zoals VS Code *\n* met een lege string en een *spatie* met een *0)*

Voorbeelden van bestanden zijn dus:

- *Puzzle.4x4:*
  0340400210030210
- *Puzzle.6x6:*
  003010560320054203206450012045040100
- *Puzzle.9x9:*
  700509001000000000150070063003904100000050000002106400390040076000000000600201004

De samurai puzzle bevat 5 regels, voor iedere regel één subsudoku. De subsudokus zijn van linksboven naar rechtsonder gedefinieerd (regel 1 = linksboven, regel 2=rechtsboven, regel 3 = midden, regel 4 = linksonder, regel 5 = rechtsonder). Let er op dat er dus overlappende velden zijn! Deze bestanden kan je hier vinden.

- *Puzzle.samurai*:

```
800000700003050206700300095000091840000007002000062000000000006090800000029030001490000000000910000000600000071200080000003404050080670000000000000070200000500030000000000000800000000400001060000503007008080000501000090000000800000000000009000600000304000000000000003908004070650000002000376000000800000001900000000009140004028000000809020000000000006100004008000000987500006700080019010607000020000090
```

Jigsaw puzzel bestand

Een Jigsaw puzzel bestand is wat complexer dan de bestanden van puzzels met reguliere vormen. Hierin moet namelijk aangegeven worden bij welk subrooster een veld hoort.

Een *puzzle.jigsaw* ziet er als volgt uit:

```
SumoCueV1=0J0=8J0=0J0=0J1=0J2=0J2=0J2=5J2=0J2=8J3=0J0=0J1=0J1=0J2=7J2=0J4=0J4=5J4=0J3=0J0=0J1=0J1=9J2=0J2=0J5=0J6=0J4=0J3=7J0=0J1=1J1=6J5=9J5=0J5=0J6=0J4=0J3=0J0=4J1=3J1=0J5=1J7=8J7=0J6=0J4=0J3=0J0=0J5=8J5=7J5=6J7=0J7=3J6=0J4=0J3=0J0=0J5=0J8=5J8=0J7=0J7=0J6=0J4=3J3=0J3=0J3=6J8=0J8=0J7=0J7=0J6=2J4=0J8=9J8=0J8=0J8=0J8=0J7=0J6=8J6=0J6
```

- Het bestand begint met *SumoCueV1*, dit kan je negeren.
- Daarna komen 81 velden zoals we gewend zijn. Maar per veld is opgenomen: *=<getal1>J<getal2>*
  - = de separator van 2 velden
  - *<getal1>* de vooraf ingevulde waarde van het betreffende veld. (0 indien leeg)
  - *J* *de scheiding tussen getal 1 en getal 2*
  - *<getal2>* *de index van het subrooster waarin dit getal zich bevindt. (tussen* 0 en 8)

Als je meer bestanden wilt opzoeken, dan kan dat hier (onderin kan je navigere naar oudere puzzels).

4. Voorbeelden van user interface

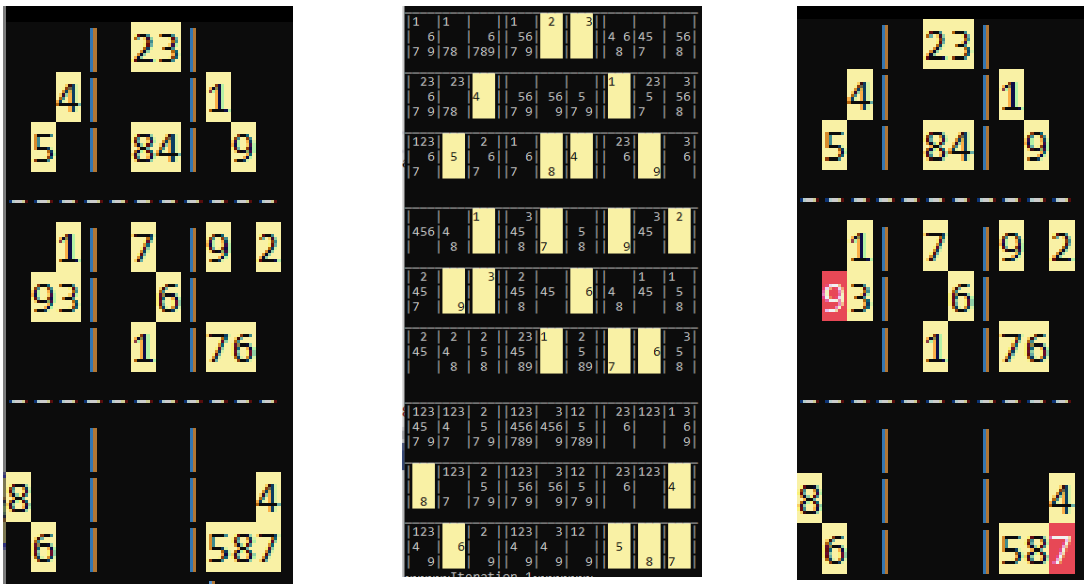
We verwachten user interface waarin interactie met de gebruiker gedaan kan worden. Dit kan een grafische user interface zijn middels bijvoorbeeld WPF of een web-frontend. Echter kan dit ook een console application zijn. Hetgeen dat we beoordelen is of er een strikte scheiding is tussen user interface in logica. *Als je deze scheiding goed hebt gedaan kost het minimale inspanning om (in de toekomst) je console interface te vervangen door een webapplicatie bijvoorbeeld!*

In het geval van een console applicatie, een paar tips:

- Gebruik hotkeys om te switchen van editorstand (bijvoorbeeld de spatiebalk)
- Gebruik de pijltjestoetsen om te navigeren tussen cellen
- Gebruik bijvoorbeeld de S voor Solve
- Gebruik bijvoorbeeld de C voor Check (valideer)
- Gebruik bijvoorbeeld een gele achtergrondkleur met witte tekst als het getal ingevuld is

- Gebruik bijvoorbeeld een rode achtergrondkleur met witte tekst als het getal foutief is.

Voorbeelden, van links naar rechts: Definitieve stand, hulpgetalstand, definitieve stand met correctie-aanduiding.



5. Backtracking algoritme

Wanneer je een puzzel oplost middels backtracking dan wil dat zeggen dat je voor elke cel een mogelijk getal probeert, de puzzel controleert of hij nog steeds valide is en vervolgens naar de volgende cel gaat. Mochten er geen mogelijke getallen meer zijn voor de laatste cel die je ingevuld hebt, dan maak je die leeg en ga je terug naar de op één na laatste cel.

Dit kan opgeschreven worden in de volgende pseudocode:

```
Function solve(board):
    cell = firstEmptyCell(board)
    If cell is not null:
        For i = 1 To 9:
            cell.Value = i
            If valid(board, cell): // check row, column and box
                If solve(board): // Recursive fill all cells after this.
                    return True // Valid option found for cell
            EndIf
        Endfor
        return False // No valid option for cell
    Else
        return True // No empty cell found
    EndIf
Endfunction
```

Bij de deze link wordt uitgelegd in twee video's hoe je dit in Python kan realiseren:

Op blackboard staat ook een *solver.py* bestand, daarin wordt een Sudoku opgelost middels backtracking. Deze kan je lokaal runnen als je python geïnstalleerd hebt, indien dat niet het geval is, dan kan je gebruik maken van Google Colaboratory, plak de code in de eerste cel en druk op play om het resultaat te zien.