

INF1010 V2013 - Obligatorisk oppgave 5 - Sudoku

Versjon 1.0

Hva er Sudoku?

I denne oppgaven skal du lage et program som løser Sudoku-oppgaver som leses fra fil, og løsningen(e) skal skrives ut ved hjelp av et grafisk brukergrensesnitt (GUI). GUIet skal programmeres slik at løsningene kan vises fram en etter en (brukeren trykker en knapp for å se neste løsning). Programmet skal også kunne skrive alle løsninger til fil. Ved hjelp av et annet grafiske brukergrensesnitt skal brukeren kunne oppgi hvilken fil oppgaven ligger på. Løsningene skal finnes ved å gå gjennom alle rutene på brettet og prøve alle mulige (lovlige) verdier i hver eneste rute. Dette kalles gjerne for en "rå kraft"-metode (greedy or brute force in English). Når en ny løsning er funnet skal den legges inn i en beholder, slik at GUI-delen som viser fram løsninger kan hente ut løsningene derfra senere. For å hjelp deg med å programmere et sudokubrett i Javas GUI, er det skrevet et [Java-program som tegner ut slike brett](#). Du kan bruke hele eller deler av dette programmet i din løsning som du vil (men du må som vanlig skrive i programmet ditt hvor du har hentet dette programmet fra), og hvis du bruker det må du forstå alle deler av det.

Sudokubrettet

Et sudokubrett består av $n \times n$ ruter. Vi bruker følgende begreper i oppgaven:

- **rute** er den minste enheten og er navnet vi bruker om de minste enhetene på brettet; feltet som det kan stå ett tall (eller en bokstav) i.
- **brett** er alle $n \times n$ ruter.
- **rad** er en vannrett (fra venstre mot høyre på brettet) rekke med n ruter.
- **kolonne** er en loddrett (ovenfra og nedover) rekke med n ruter.
- **boks** er flere vannrette og loddrette ruter, markert med tykkere strek i oppgavene; i 9x9-eksemplet er en boks på 3x3 ruter, mens linsudoku består en boks av 2x3 ruter.

Du skal lage programmet så generelt at det kan løse sudokubrett som ikke har kvadratiske bokser, for eksempel såkalte linsudokuer på 6x6 ruter som har bokser på 2x3 ruter og som du finner nær slutten av Aftenpostens kulturdel (anbefales for nybegynnere). Husk også at 9x9 ikke er noen øvre grense for størrelsen på brettet.

I denne obligatoriske oppgaven er en sudokuoppgave et delvis utfylt brett som kan ha tre løsningsmuligheter:

1. En løsning (slike finner vi i aviser, bøker og blader)
2. Ingen løsning (tallene er plassert slik at det ikke finnes en løsning)
3. Flere løsninger (for få forhåndsutfylte tall)

Hint: Under utviklingen kan det være lurt å først lage et program som genererer alle løsninger for et tomt brett (brett- og boksstørrelse eneste inndata), for så senere å utvide med at noen av rutene kan ha forhåndsutfylte verdier. Ikke bruk et tomt brett med mer enn 9 x 9 ruter, da dette kan ta **fryktelig lang tid**.

Det er main-metoden i programmet ditt som skal starte med å finne alle løsningene. Main-tråden skal opprette et eller flere vinduer for å lese inn data. Disse vinduene skal så lukkes når data er lest inn. I denne oppgaven trenger du ikke programmere med andre tråder enn main-tråden og GUI-tråden. Når main-tråden har funnet en løsning legges denne inn i en beholder, og når alle løsningene er funnet skal disse skrives ut en etter en av et GUI som først opprettes når alle (eller 500) løsninger er funnet. Prøv å lage et robust program, dvs. et som ikke kræsjer når filformatet er feil eller noe annet uventet skjer.

Besvarelsen din SKAL følge disse retningslinjene og den SKAL inneholde disse delene:

Programmet SKAL inneholde "class Sudokubeholder" som igjen inneholder de tre offentlige metodene settInn (eventuelt insert hvis du foretrekker engelske navn på identifikatorer) taUt (get), og hentAntallLosninger (getSolutionCount). Du kan gjerne bruke ferdiglagde datastrukturer i Java-biblioteket når du programmerer denne klassen.

Den første delen av programmet (som utføres av main-tråden) skal finne løsninger og legge dem inn i et objekt av klassen Sudokubeholder. Når alle løsninger er funnet skal main metoden starte opp et grafisk brukergrensesnitt der brukeren kan be om at en og en løsning blir vist fram. Hvis det finnes flere løsninger enn det er plass til i Sudokubeholderen, skal bufferet holde orden på hvor mange løsninger som er funnet, men ikke ta vare på flere løsninger etter at beholderen er full (500 stykker).

Programmet ditt SKAL inneholde klassene Rute (Square) og Brett (Board). Klassen Brett SKAL inneholde en todimensjonal tabell som peker ut alle rutene. Klassen Rute SKAL ha to subclasser, en for ruter som har en forhåndsutfylt verdi, og en for ruter der du skal finne en mulig verdi.

I tillegg SKAL du ha tre klasser som du kaller Boks (Box), Kolonne (Column) og Rad (Row). Du skal lage ett objekt av disse klassene for hver rad, kolonne og boks på brettet. Disse tre klassene SKAL ha en felles superklasse, og subclassene skal gjenbruke mest mulig av koden fra superklassen. Når en rute sjekker om den kan bruke en verdi, SKAL ruten kalle metoder som gjør dette i rutens Kolonne-objekt, Rad-objekt og Boks-objekt. Klassen Rute SKAL IKKE ha noen datastruktur (annet enn sin egen verdi) som sier noe om hvilke verdier som er lovlig i denne ruten. Dette siste "SKAL IKKE" kravet kan du fravike under gitte betingelser, se eget avsnitt lenger nede om dette.

Hver enkelt rute SKAL ha en metode, fyllUtRestenAvBrettet (fillInnRemainingOfBoard), som prøver å sette alle tall i seg selv (den prøver først med 1, så 2, så 3 osv.), og lykkes dette for et tall, kalles samme metode (fyllUtRestenAvBrettet) i neste rute (dvs den rett til høyre). Når en vannrett rad er ferdig (det finnes ingen rute rett til høyre), kalles metoden i ruten helt til venstre i neste rad, osv. Når et kall på fyllUtRestenAvBrettet-metoden i neste

rute returnerer, prøver ruten neste tall som enda ikke er prøvd, osv. helt til alle tall er prøvd i denne ruten. Main-metoden starter det hele ved å kalle `fillUtRestenAvBrettet` i den øverste venstre ruten. (Hint: Du kan gjerne lenke sammen alle rutene med en neste-peker, slik at en rute bare kan kalle `neste.fillUtRestenAvBrettet`). Når metoden `fillUtRestenAvBrettet` har funnet en lovlig verdi i den siste ruten (den nederst til høyre) på brettet, legges denne løsningen inn i beholderen.

Størrelsen på brettet, størrelsen på boksene og de sifrene som alt er fylt inn SKAL leses fra fil. Programmet SKAL kunne bruk `JFileChooser` til å lese inn filnavn, og filformatet skal være slik som beskrevet under (gruppelæren som skal godkjenne oppgaven din vil teste programmet ditt med sine egne filer). Hvis det oppgis et filnavn som parameter til programmet (på kommandolinja) skal oppgaven leses fra denne filen (istedenfor vha. `JFileChooser`). Hvis det oppgis to filnavn skal oppgaven løses fra den første filen, og løsningene skrives på den andre filen (og ikke vises frem ved hjelp av GUI). Løsningene skal skrives på fil som vist under.

Om du synes at noen av disse kravene er urimelige, eller du synes du kan løse oppgaven mer elegant eller bedre på en annen måte, så snakk med gruppelæreren som skal rette oppgaven din. Hvis du får skriftelig (på epost) tillatelse så kan du løse oppgaven på en annen måte.

Om å fravike "SKAL IKKE"-kravet om datastrukturen i klassen Rute.

Det er i orden å ha konstante verdier i klassen `Rute` som sier noe om gyldige verdier i denne ruten. Slike konstante verdier vil være de forhåndsutfylte verdiene i samme rad, kolonne eller boks. Hvis du ønsker å ha variable som kan brukes til å midlertidig restrikt ytterligere gyldige verdier i denne ruten, må du først diskutere dette med gruppelæreren som skal rette oppgaven din, du må sende ham eller henne en epost om dette, og du må få skriftelig tillatelse på epost fra gruppelæreren.

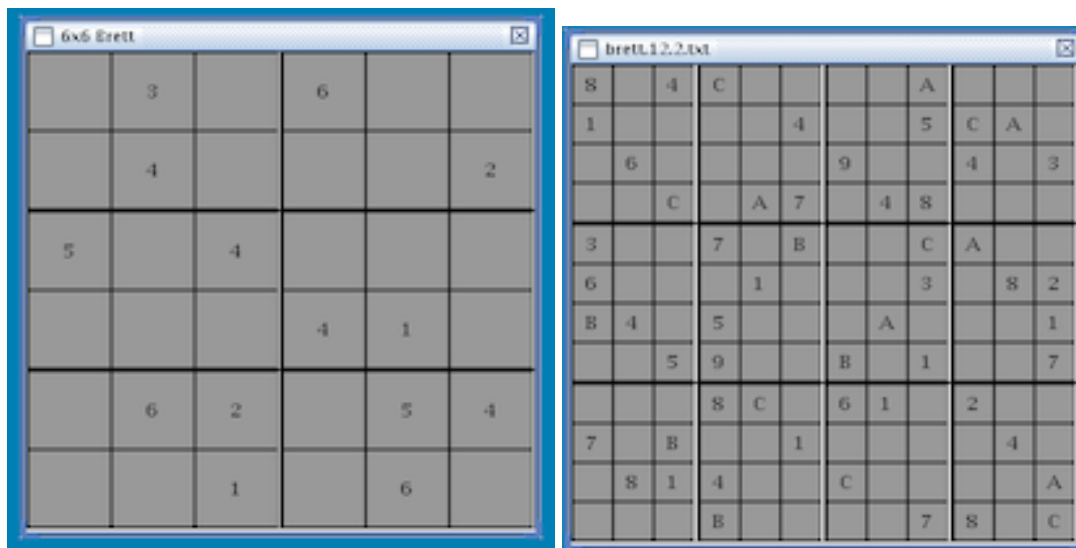
Tilbud om hjelp underveis:

Du bør jevnlig diskutere med gruppelæreren din hvordan du skal løse denne oppgaven. Pass på å hele tiden ha et program som kompilerer og kjører (men som i starten ikke gjør særlig mye). Du bør i alle fall kontakte gruppelæreren din for å få en tilbakemelding:

- Når du har laget main og mange tomme klasser og har en første grove skisse av hele programmet ditt
- Når du har bestemt formatet på løsningene slik de skal lagres i `Sudokubeholderen`
- Når du har laget en skisse av klassen `Rute` og dens subclasser
- Når du har laget en skisse av klassene `Boks`, `Kolonne` og `Rad` og superklassen til disse klassene.
- Når du har laget en skisse av GUI-programmet som henter ut løsninger fra `Sudokubeholderen` og tegner dem ut.

Filformat

Under ser du to brett, 6x6 og 12x12, som begge har én løsning. Klikker du på bildet ser du hvordan filformatet ser ut. Legg spesielt merke til at tall > 9 representeres med store bokstaver.



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

8		4	C				A				
1				4			5	C	A		
	6					9		4		3	
		C		A	7		4	8			
3			7		B			C	A		
6				1				3		8	2
B	4		5				A				1
		5	9			B	1				7
			8	C		6	1		2		
7		B			1					4	
	8	1	4			C					A
			B				7	8		C	

Her er en [oppgave](#) (6x6-brett) som har [28 løsninger](#). Dette kan du bruke for å teste ditt eget program. De 28 løsningene er skrevet på filformatet du skal bruke for å skrive løsninger. I [denne mappa](#) finnes flere delvis utfylte brett. Noen av brettene har flere løsninger. (OBS. Filen 28losninger6x6.txt inneholder ikke en oppgave, men løsninger). Hvis du vil kan du anta at største brett er 16 x 16 ruter.

Hvis du ønsker flere utfordringer

De som ønsker en utfordring kan gjerne gjøre det mulig å vise frem løsninger etter som de blir ferdige, og hvis det finnes flere løsninger enn det er plass til i beholderen, kan nye løsninger finnes ettersom det blir plass i beholderen når gamle løsninger vises fram. Pass på at det ikke er GUI-tråden som arbeider med å finne løsninger. Du kan også lage GUIet slik at du i tillegg kan legge inn oppgaver ved å taste inn tall (og bokstaver) i et tomt brett. Trenger du enda flere utfordringer kan du prøve å lage et GUI som kan hjelpe deg med å løse Sudoku-oppgaver.