

# SI-pass 6 OOP

**Handledare:** Oscar Söderlund

**Mail:** soscar@student.chalmers.se

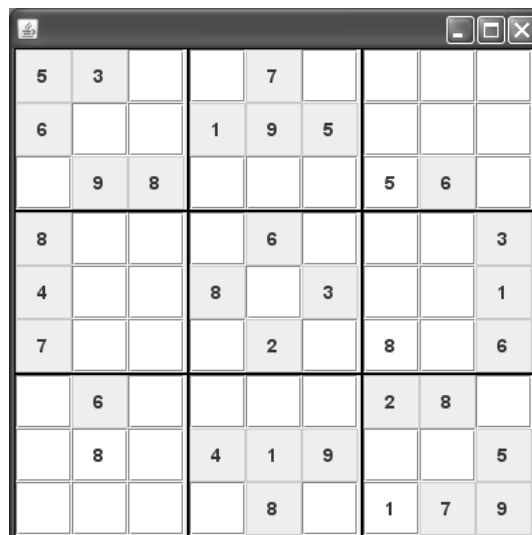
1. I Sudoku använder man normalt en spelplan som består av 9x9 **rutor** vilka kan innehålla siffrorna 1-9. Spelplanen är indelad i 9 s.k. regioner bestående av 3 rader och 3 kolumner, så som framgår av figuren.

Från början är några av rutorna ifyllda med siffror och kan inte ändras. Dessa är gråa i figuren. Det gäller att fylla i de övriga rutorna på så sätt att varje rad, kolumn och region kommer att innehålla siffrorna 1-9 och så att varje siffra bara förekommer en gång i en viss rad, kolumn eller region.

I denna uppgift är målet att konstruera ett program som låter användaren spela Sudoku på datorn. När spelet börjar skall programmet visa ett fönster där de givna siffrorna redan är ifyllda. För att uppgiften skall bli lite mer lätthanterlig har den delats i ett antal deluppgifter vilka kan lösas *helt oberoende av varandra*. (Du behöver alltså inte ha löst en viss deluppgift för att få använda dig av den i en annan deluppgift.)

Följande klass är given och *skall* användas i deluppgifterna:

```
class Ruta extends JTextField {
    private int rad, kol; // rutans plats
    private char tecken; // tecken som skall visas
    public Ruta(int r, int k, char c) {
        rad = r;
        kol = k;
        setHorizontalAlignment(JTextField.CENTER);
        setFont(new Font("SansSerif", Font.BOLD, 14));
        setTecken(c);
    }
    public int getRad() { return rad; }
    public int getKol() { return kol; }
    public char getTecken() { return tecken; }
    public void setTecken(char c) {
        tecken = c;
        setText("" + c);
    }
}
```



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

- (a) Skriv ett program som visar en spelplan enligt figuren på föregående sida. Fönstrets storlek skall vara 400x400 pixlar. I denna deluppgift skall varje **Ruta** i spelplanen vara blank. (Det skall alltså inte ännu visas några siffror i rutorna.) Klassen som programmet ligger i skall heta **Sudo**.

Deklarera i klassen **Sudo** en instansvariabel med namnet **rutor** vilken skall vara ett tvådimensionellt fält med 9 rader och 9 kolumner, där komponenterna är av typen **Ruta**. I konstruktorn till klassen **Sudo** skall variabeln **rutor** initieras och det skall finnas kod som genererar ett fönster som ser ut som i figuren. Det skall finnas gränslinjer i fönstret så att man tydligt ser de nio regionerna, precis som i figuren.

Resten av klassen **Sudo** beskrivs i de följande deluppgifterna och behöver inte skrivas ännu.

(8p)

- (b) Skriv i klassen **Sudo** en privat instansmetod med namnet **nyttSpel**. Metoden skall vara så utformad att man kan anropa den varje gång man vill påbörja ett nytt spel.

Du kan t.ex. tänka dig att användaren har valt alternativet Nytt spel på en meny. Du kan också anta att denna metod anropas sist i konstruktorn i klassen **Sudo**. Metoden **nyttSpel** skall fylla i alla rutorna i spelplanen **rutor**. Först skall den i en dialogruta fråga efter namnet på en textfil. Filen förväntas innehålla de givna siffrorna. De **rutor** som skall vara blanka från början markeras i filen med nollor. Spelplanen i figuren ovan representeras t.ex. av en fil med innehållet:

```
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

De **rutor** som markeras med nollor skall i spelplanen innehålla ett blankt tecken och vara ändringsbara. De övriga rutorna skall visa den rätta siffran och vara icke-ändringsbara (Tips. metoden **setEditable**).

Du behöver inte kontrollera att den fil användaren anger finns. (Låt det bli en exception om den inte finns.) Däremot skall du kontrollera att filen bara innehåller enstaka siffror (dvs. tal i intervallet 1 till 9 med blanka tecken mellan) och att det finns minst 81 siffror i filen. Skulle så inte vara fallet skall en dialogruta med ett felmeddelande visas och programmet avbrytas.

(8p)

- (c) Skriv i klassen **Sudo** en privat instansmetod med namnet **OK**. Metoden skall användas för att undersöka om det går att lägga in ett viss tecken i en viss **Ruta**. Metoden skall ha tre parametrar: en **char** som innehåller det tecken man försöker lägga in och två int-parametrar vilka anger i vilken rad resp. kolumn man vill lägga in tecknet. Metoden skall ge ett returvärde av typen boolean.

Ett blankt tecken skall alltid ge värdet **true**. Om tecknet inte är blankt och inte är en siffra i intervallet '1' till '9' skall metoden ge värdet **false**. Om tecknet är en siffra i intervallet '1' till '9' skall metoden, genom att undersöka fältet **rutor**, se om det går bra att lägga in siffran i den angivna rutan. (Den aktuella siffran får då inte redan finnas i den rad, kolumn eller region det gäller.) Går det bra skall värdet **true** returneras annars returneras värdet **false**.

(8p)

- (d) Det sista steget är att lägga till en hanterare så att användaren kan skriva in siffror i de tomma rutorna. Uppgiften är att skriva en hanterare som tar hand om händelser av typen **KeyReleased**. (Anledningen till att man inte skall använda **ActionPerformed** är att användaren då skulle varit tvungen att trycka på Entertangenten efter varje inmatad siffra.)

I hanteraren skall man använda metoden **getKeyChar** för att få reda på vilket tecken användaren skrivit in. Om det går bra att lägga in detta tecken i den **Ruta** händelsen inträffade i (testa med metoden **OK**) så skall det inmatade tecknet läggas in i och visas i rutan. Om det inte går att lägga in det inmatade tecknet skall istället ett blankt tecken läggas in och visas i rutan.

Programmet kommer i och med detta att förhindra att användaren skriver in siffror så att det blir dubbletter i någon rad-, kolumn eller region. Visa också här vilken kod som måste läggas in i konstruktorn i klassen **Sudo** för att man skall komma till hanteraren när användaren skriver något i en **Ruta**.

(6p)

(Skansholm 2009)