INF1000 - Notat om I/O i Java

Tuva Kristine Thoresen, Ingrid Grønlie Guren
tuvakt@ulrik.uio.no, ingridgg@ulrik.uio.no

22. oktober 2014

Dette notatet handler om I/O (input/output) i Java, og tar for seg innlesning fra terminal, lesing fra fil og skriving til fil.

1 Innlesning fra terminal

Innlesing fra terminal bruker vi ofte i interaksjon med brukere av programmet. Programmet ber om input, som brukeren så taster inn. Programmet må deretter lese inn det som blir skrevet og bearbeide informasjonen.

For å få til innlesning fra terminal bruker vi en ferdiglaget Java-klasse som heter `Scanner`. Denne klassen inneholder mange nyttige funksjoner for innlesning.

For å bruke denne klassen må vi først importere de nødvendige programmene. Disse ligger i `java.util` og kan importeres på følgende måte:

```
import java.util.Scanner;
```

Neste steg er å opprette et innlesningsobjekt. Det må vi gjøre slik at vi får tak i hele `Scanner`-klassen og dens funksjonaliteter. Dette gjøres slik:

```
Scanner in = new Scanner(System.in);
```

Når vi leser fra terminalen leser vi fra standard input. Dette får vi tak i gjennom `System.in`. Legg merke til at dette er nesten helt likt som når vi printer til terminalen, nemlig med `System.out`.

1.1 Innlesning av tekst

`Scanner` har en metode som vi bruker for å lese inn tekst:

- `nextLine()` - leser inn en linje av tekst.

Denne metoden kan brukes på følgende måte:

```
System.out.println("Tast inn en setning:");
String linje = in.nextLine();
```

1.2 Innlesning av tall

For å lese inn tall med `Scanner` leser vi først inn en linje, slik vi gjør når vi leser inn tekst. Så konverterer vi denne linja til et tall. Dette kan skje på følgende måte:

```
String linje = in.nextLine();
int heltall = Integer.parseInt(linje);
```

Her konverterer vi teksten `linje` til et heltall ved kall på metoden `Integer.parseInt()`. Denne metoden tar en tekst som inneholder et tall (for eksempel "4") som input og konverterer denne til et heltall (her med verdi 4). På tilsvarende måte kan man konvertere en tekst til et desmialtall, med metoden `Double.parseDouble()`. Eksempel på innlesning av heltall:

```
System.out.println("Tast inn et heltall:");
String linje = in.nextLine();
int heltall = Integer.parseInt(linje);
```

1.3 Eksempelprogram

Nå skal vi lage et lite program som leser inn en setning, et heltall og et desimaltall fra brukeren.

```
import java.util.Scanner;

class InputAlt {
    public static void main(String[] args) {

        // oppretter innlesningsobjekt:
        Scanner in = new Scanner(System.in);

         // leser inn en setning
        System.out.println("Tast inn ditt fulle navn:");
        String navn = in.nextLine();

        // leser inn et heltall:
        System.out.println("Tast inn et heltall:");
        String linje = in.nextLine();
        int heltall = Integer.parseInt(linje);
```

```
        // leser inn et desimaltall:
        System.out.println("Tast inn et desimaltall:");
        String tekst = in.nextLine();
        double desimaltall = Double.parseDouble(tekst);
    }
}
```

2 Innlesning fra fil

Når vi leser inn fra terminalen med `Scanner` leser vi fra `System.in`. For å lese inn fra fil, kan vi også bruke `Scanner`. Forskjellen er at nå må vi sende med filen som argument til `Scanner`.

Det er dessverre noen små problemer med å lese fra fil. Vi har nemlig ingen garanti for at filen vi prøver å lese fra, faktisk finnes! Java lar oss løse dette ved at vi først prøver å lese fra filen. Hvis dette går galt, får vi en feilmelding som vi må håndtere.

Vi må først huske å importere de nødvendige programmene:

```
import java.util.Scanner;
import java.io.*;
```

og så kan vi begynne å lese. Vi oppretter først et innlesningsobjekt, og forteller det at det skal lese fra filen *innfil.txt*:

```
String filnavn = "innfil.txt";
File fil = new File(filnavn);
Scanner innFil = new Scanner(fil);
```

2.1 Exceptions

Når vi leser inn en fil, vet ikke Java om filen eksisterer. Derfor er vi nødt til å fortelle hva programmet skal gjøre dersom innlesingen fra fil ikke går. Her bruker vi `Exceptions` (på norsk: unntak).

Hvis en metode skal lese fra fil, må metoden *kaste et unntak* hvis noe går galt. Dette gjør vi for eksempel slik:

```
public static void main(String[] args) throws Exception {
    // Kode for innlesing av fil
}
```

Noen ganger har vi lyst til å skrive en egen metode for å lese inn fra fil. Da er det viktig at denne metoden kaster et unntak:

```
public static void lesFil() throws Exception {
    // Kode for innlesning av fil
}
```

Når en metode kaster et unntak er vi helt avhengige av at alle metoder som kaller denne metoden også kaster et unntak. Hvis metoden `lesFil()` kalles fra main-metoden vår, må vi huske å kaste et unntak der også:

```
public static void main(String[] args) throws Exception {
    // Metode for innlesing av fil
    lesFil();
}
```

2.2 Innlesning av en typisk fil

Innlesning fra fil fungerer på samme måte som innlesning fra terminal. Vi kan bruke de samme `Scanner`-metodene for å lese inn informasjon:

```
// Leser inn en linje
String linje = innFil.nextLine();

// Leser inn et heltall
String s = innFil.nextLine();
int tall = Integer.parseInt(s);
```

Men, ofte har filer en bestemt struktur som gjentar seg. La oss se på en fil på et slikt format:

```
Ola
Tuva
Heidi
Lars
Gunnar
Ingrid
.....
```

Her har vi en fil med flere navn, et navn på hver linje. Vi vet ikke hvor mange linjer fila vår har - hvordan kan vi da lese fra den?

Løsningen ligger i en metode i `Scanner`:

- `hasNextLine()`: Denne metoden returnerer `true` hvis det finnes flere linjer i fila, `false` ellers.

For å lese inn hele filen, kan vi da gjøre følgende:

```
while (innFil.hasNextLine()) {
  String linje = innFil.nextLine();
}
```

3 Skrive til fil

For å skrive til fil bruker vi Java-klassen `PrintWriter` og metoden `println`:

```
PrintWriter writer = new PrintWriter("utfil.txt");

// skrive til filen
writer.println("her er linje 1");
writer.println("og her kommer linje 2");

// lukke filen
writer.close();
```

Her skriver vi til en fil ved navn "utfil.txt". Hvis denne filen ikke finnes fra før, blir den opprettet. Hvis ikke blir det gamle innholdet i filen overskrevet, og ersattet med det nye du skrev. Det er viktig å lukke skriveren når du er ferdig, uten dette blir ikke filen skrevet til. Dette gjør du med metoden `writer.close()`.

`PrintWriter` må importeres fra `java.io`:

```
import java.io.*;
```

Metoden som skriver til fil må kaste et unntak, `Exception` akkurat slik vi må gjøre når vi skriver til fil:

```
public void skrivTilFil() throws Exception {
    // Kode for skriving til fil
}
```

3.1 Et eksempel

I dette eksempelet vil vi skrive alle tall mellom 0 og 9 til en fil.

```
import java.io.*;

class Skriveeksempel {
    public static void main(String[] args) throws Exception {
        skrivTilFil();
    }

    public void skrivTilFil() throws Exception {
        PrintWriter writer = new PrintWriter("utfil.txt");

        // skrive
        for (int i = 0; i < 10; i++) {
            writer.println(i);
        }

        // lukke
        writer.close();
    }
}
```