

Klassen IO

Class `IO` er en Java-klasse til simpel terminalorienteret indlæsning og udskrivning. Formålet med klassen er at give en Java-programmør let adgang til indlæsning fra tastatur og formateret udskrivning på skærm.

Her er et eksempelprogram, der illustrerer brugen af class `IO`. Programmet indlæser en række heltal fra tastaturet og udskriver til slut deres sum.

```
import IO.*;

public class Adder {
    public static void main(String args[]) {
        int Sum = 0;
        while (!IO.eof())
            Sum += IO.readInt();
        IO.println(Sum);
    }
}
```

Et tilsvarende program, der benytter sig af Javas indbyggede klasser, kan for eksempel se således ud:

```
import java.io.*;

public class Adder{
    public static void main(String args[]) {
        int Sum = 0;
        StreamTokenizer input = new StreamTokenizer(System.in);
        try {
            while (input.nextToken() != StreamTokenizer.TT_EOF)
                Sum += input.nval;
        }
        catch (IOException e) {}
        System.out.println(Sum);
    }
}
```

Udviklingen af dette sidste program kræver godt kendskab til Javas undtagelsesbegreb (exceptions) og til klassen `StreamTokenizer`. Det første program er derimod er let at skrive.

Et andet problem, som class `IO` løser, er Javas mangel på faciliteter til formateret udskrivning. Antag for eksempel, at der ønskes udskrevet en tabel over den naturlige logaritmefunktion, $\ln(x)$, for $x = 0.25, 0.5, 1, \dots, 2.5$. Et fragment af et Java-program, der løser denne opgave, ville være følgende:

```
for (int i = 1; i <= 10; i++) {
    double x = i/4.0;
    System.out.println(x + " " + Math.log(x));
}
```

Ved en kørsel fås imidlertid følgende noget uoverskuelige udskrift:

```
0.25 -1.38629
0.5 -0.693147
0.75 -0.287682
1 0
1.25 0.223144
1.5 0.4054
1.75 0.559616
2 0.693147
2.25 0.81093
2.5 0.916291
```

En bedre opstilling af tabellen fås ved anvendelse af udskrivningsmetoden `print` i class `IO`. Et kald `print(x,n,w)` bevirker, at tallet `x` udskrives højrejusteret med `n` decimaler i et tekstfelt på `w` tegn. Nedenfor er tabeludskrivningen foretaget ved hjælp af denne metode.

```
for (int i = 1; i <= 10; i++) {
    double x = i/4.0;
    IO.print(x,2,6);
    IO.println(Math.log(x),5,10);
}
```

Udskriften blev følgende:

```
0.25 -1.38629
0.50 0.69315
0.75 0.28768
1.00 0.00000
1.25 0.22314
1.50 0.40547
1.75 0.55962
2.00 0.69315
2.25 0.81093
2.50 0.91629
```

Her følger en kort oversigt over metoderne i class `IO`.

Metoder til brug ved indlæsning:

eof():

Returnerer `true`, når der ikke kan læses flere symboler fra standard input (`System.in`). Ellers `false`.

**readBoolean(),
readDouble(),
readFloat(),
readInt(),
readLong(),
readToken():**

Indlæser det næste symbol (`Boolean`, `Double`, `Float`, `Int`, `Long` eller `Token`) fra standard input og returnerer den tilhørende værdi som resultat. Hvis der ikke kan læses flere symboler, udskrives en fejlmeddelelse, og programmet terminerer.

Metoder til brug ved udskrivning:

print(x):

Et kald svarer til `System.out.print(x)`.

**print(int x, int w),
print(long x, int w):**

Udskriver heltallet `x` højrejusteret i et tekstfelt af bredde `w` tegn. Såfremt feltet ikke kan rumme tallet, udskrives det i et felt bestående af det nødvendige antal tegn.

**print(float x, int n, int w),
print(double x, int n, int w):**

Udskriver det flydende tal `x` med `n` decimaler højrejusteret i et tekstfelt af bredde `w` tegn. Såfremt feltet ikke kan rumme tallet, udskrives det i et felt bestående af det nødvendige antal tegn.

**printReal(float x, int n, int w),
printReal(double x, int n, int w):**

Udskriver det flydende tal `x` i "videnskabelig notation" med `n` betydende cifre og højrejusteret i et tekstfelt af bredde `w` tegn. Såfremt feltet ikke kan rumme tallet, udskrives det i et felt bestående af det nødvendige antal tegn.

Tallet udskrives med et ciffer mellem 1 og 9 (inklusive) foran punktummet efterfulgt af n-1 decimaler og en eksponentdel. For tal af typen `float` indeholder eksponentdelen 2 cifre, mens den for tal af typen `double` indeholder 3 cifre.

Eksempel. Hvis tabeludskrivningsprogrammet ændres til

```
IO.printReal(x,4,12);
IO.printlnReal(Math.log(x),6,20);
```

fås følgende udskrift

2.500e-001	-1.38629e+000
5.000e-001	-6.93147e-001
7.500e-001	-2.87682e-001
1.000e+000	0.00000e-001
1.250e+000	2.23144e-001
1.500e+000	4.54650e-001
1.750e+000	5.59616e-001
2.000e+000	6.93147e-001
2.250e+000	8.10930e-001
2.500e+000	9.16291e-001

```
println(...),
printlnReal(...):
```

Udskriver som `print()` og `printReal()`, idet der dog afsluttes med et linjeskift.

```
flush():
```

Har samme virkning som `System.out.flush()`.