Brugerinput i Java

Denne her artikel gennemgår diverse ting ved brug af brugerinput i Java. Den starter med det simple og fortæller derefter skridt for skridt om diverse gode ting, man kan bruge brugerinput til.

Skrevet den **05. Feb 2009** af **ttn-** I kategorien **Programmering / Java** | ★★☆☆☆

Brugerinput i Java.

Nu vil vi se lidt på simpel brug af brugerinput i Java. De fleste har nok prøvet, at de gerne vil have, at brugeren af programmet skal kunne indtaste egne oplysninger, der så bliver behandlet og derefter bliver vist som output til brugeren. Til alle, der har prøvet c++, ved de, at det i C++ er rimelig let, hvilket desværre ikke er helt så let i Java - selvom det da heller ikke er kompliceret.

Jeg vil nu vise et eksempel, hvor man kan have brug for brugerinput:

udenBrugerInput:

```
//Opretter variablene
String ditFornavn = "Mikael";
String ditEfternavn = "Milhoj";
int dinAlder = 15;

//Udskriver
System.out.println("Hej " + ditFornavn + " " + ditEfternavn + ".");
System.out.println("Du er " + dinAlder + " aar gammel!");
```

Da det jo ikke er alle, der hedder det samme som mig, vil det være hensigtsmæssigt at lade brugeren indtaste ditFornavn, ditEfternavn og dinAlder.

For at kunne bruge brugerinput i Java, skal man bruge klassen BufferedReader fra pakken "java.io".

medBrugerInput:

```
import java.io.*; //importerer packagen java.io.

public class medBrugerInput {
    public static void main(String[] args) throws IOException {

        //Opretter en instans af klassen BufferReader, så vi kan modtage input
fra brugeren
        BufferedReader input = new BufferedReader(new
InputStreamReader(System.in));
```

```
        //Udskriver, hvad brugeren skal skrive af input
        //Og gemmer disse
        System.out.println("Skriv venligst dit fornavn:");
        String ditFornavn = input.readLine();

        System.out.println("Skriv venligst dit efternavn:");
        String ditEfternavn = input.readLine();

        System.out.println("Skriv venligst din alder");
        String dinAlder = input.readLine();


        //Udskriver
        System.out.println("Hej " + ditFornavn + " " + ditEfternavn + ".");
        System.out.println("Du er " + dinAlder + " aar gammel!");
    }
}
```

Jeg vil nu komme kort ind på, hvad de enkelte nye linier i koden gør.

"import java.io.*;" importerer pakken IO, så vi kan gøre brug af diverse metoder, der kan læse eller skrive.
" throws IOException" kaster en IOException, der skal kastes, hvis man have brugerinput. Så hvis brugeren laver en fejl, koden ikke kan lide, vil der blive kastet en exception.

"BufferedReader input = new BufferedReader(new InputStreamReader(System.in));"
System.in er det modsatte af System.out, der bruges til at lave output med, sjovt nok. System.in er en binær stream. Derfor bruger man InputStreamReader til at gøre inputtet til tegn. Tegnene bliver så lavet om til en linie med BufferdReader.

"input.readLine();" Dette er her, hvor brugeren får lov at indtaste sit input. Derefter bliver brugerens input så gemt i en String, der så bliver udskrevet.

Som den opmærksomme læser sandsynligvis har lagt mærke til, så er alderen ikke en int længere, men nu en string. Det er fordi, at inputtet jo er tekstbaseret, hvorfor int ikke er brugbart. Men i nogle situationer kan det jo alligevel være rart at have det som en variable i stedet for, da man måske skal bruge det til fx. at lægge tal sammen.

simpelLommeregner version 1:

```
//Opretter en instans af klassen BufferReader, så vi kan modtage input fra
brugeren
BufferedReader input = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Velkommen til den simple lommeregner);
```

```
System.out.println("Jeg kan laegge to tal sammen!");

//Udskriver hvad brugeren skal skrive
//Derefter bliver det gemt i en String
System.out.println("Skriv det første tal her:");
String tal1 = input.readLine();

System.out.println("Skriv det andet tal her:");
String tal2 = input.readLine();
Double result = tal1 + tal2;

System.out.println("Resultatet er: " + result);
```

Hvis man compiler dette, står der, at den ikke kan konvertere en String til en double på den måde. Derfor skal man konvertere ens Strings til fx. doubles, FØR man lægger tallene sammen. Der er skrevet om, hvordan du konverterer fra String til int osv. i en artikel lavet af Arne_v. Den er værd at læse.

simpelLommeregner version 2:

```
//Opretter en instans af klassen BufferReader, så vi kan modtage input fra
brugeren
BufferedReader input = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Velkommen til den simple lommeregner");
System.out.println("Jeg kan laegge to tal sammen!");

//Udskriver hvad brugeren skal skrive
//Derefter bliver det gemt i en String
System.out.println("Skriv det første tal her:");
String tal1 = input.readLine();
double doubleTal1 = Double.parseDouble(tal1);

System.out.println("Skriv det andet tal her:");
String tal2 = input.readLine();
double doubleTal2 = Double.parseDouble(tal2);

//Finder resultatet at de to doubles lagt sammen!
double result = doubleTal1 + doubleTal2;

//Udskriver resultat!
System.out.println("Resultatet er: " + result);
```

Også har vi så lavet en simpel lommeregner, der kan lægge tal sammen. Men hov! Hvis man pludselig skriver abcd i stedet for et tal, får man en **"NumberFormatException"**. Problemet opstår ved, at en String godt kan modtage bogstaver og symboler, hvilket gør det muligt, mens double kun kan bruge tal. Så derfor bliver vi nødt til at kaste en exception. Hvis du ikke ved, hvad en exception er, er det en god idé at læse om det.

simpelLommeregner version 3:

```
//Opretter en instans af klassen BufferReader, så vi kan modtage input fra
brugeren
BufferedReader input = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Velkommen til den simple lommeregner");
System.out.println("Jeg kan laegge to tal sammen!");

//Udskriver hvad brugeren skal skrive
//Derefter bliver det gemt i en String
try {
    System.out.println("Skriv det første tal her:");
    String tal1 = input.readLine();
    double doubleTal1 = Double.parseDouble(tal1);

    System.out.println("Skriv det andet tal her:");
    String tal2 = input.readLine();
    double doubleTal2 = Double.parseDouble(tal2);

    //Finder resultatet at de to doubles lagt sammen!
    double result = doubleTal1 + doubleTal2;

    //Udskriver resultat!
    System.out.println("Resultatet er: " + result);

} catch(NumberFormatException e) {
    System.out.println("NumberFormatException blev fanget. Programmet må
lukke");
}
```

Nu prøver vi at fange en NumberFormatException. Hvis den bliver fanget, udskriver vi, at der blev fanget en exception og programmet må lukke. Man kan også gøre sådan, at programmer vil starte forfra, hvis der bliver fanget en NumberFormatException. Jeg kan én metode til at gøre dette og det er ved at putte det ind i en do.. while løkke - om der findes andre og bedre måder at gøre det på, ved jeg ikke - hvis der gør, så må I endelig skrive det til mig på mom33@hotmail.com, så vil jeg tilføje det til artiklen.

simpelLommeregner version 4:

```
//Opretter en boolean ved navn exception og gør den til falsk!
boolean exception = false;

//Opretter en instans af klassen BufferReader, så vi kan modtage input fra
brugeren
BufferedReader input = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Velkommen til den simple lommeregner");
System.out.println("Jeg kan laegge to tal sammen!");
```

```
//Vi bruger til at tjekke, om exception = true
//hvis den er det, så vil programmet starte forfra
//Hvis den er false, så stoppet programmet

do {
    try {
        //Udskriver hvad brugeren skal skrive
        //Derefter bliver det gemt i en String
        System.out.println("Skriv det første tal her:");
        String tal1 = input.readLine();
        double doubleTal1 = Double.parseDouble(tal1);

        System.out.println("Skriv det andet tal her:");
        String tal2 = input.readLine();
        double doubleTal2 = Double.parseDouble(tal2);

        //Finder resultatet at de to doubles lagt sammen!
        double result = doubleTal1 + doubleTal2;

        //Udskriver resultat!
        System.out.println("Resultatet er: " + result);

        //Prøver at fange en NumberFormatException
    } catch(NumberFormatException e) {
        //Hvis der bliver kastet en exception, bliver boolean exception = true
        exception = true;
        System.out.println("NumberFormatException blev fanget.");
                System.out.println("Programmet genstartes!");
    }

//Hvis exception == true, starter programmet om igen. Ellers lukker det!
} while (exception == true);
```

Vi starter nu med at oprette en boolean ved navn exception, som vi sætter til false. Denne skal vi bruge senere. Derefter laver vi en do..while løkke rundt om hele try...catch blokken. Grunden til, at man skal bruge do...while løkken, er, at man er sikret, at programmet kører mindst en gang. Do...while løkkens argument er, at hvis exception = true, skal programmet starte forfra. Hvis en NumberFormatException bliver fanget, vil exception blive lavet om til true. Ellers vil den fortsætte med at være false og gennemløbet af do..while løkke, og derfor også programmet, vil stoppe, da exception jo ikke er sand.

Den sidste ting, jeg vil fortælle om, er muligheden for at lade programmet fortsætte. I stedet for at lade programmet stoppe, kan man så give brugeren muligheden for at fortsætte eller stoppe programmet. Det er især praktisk, hvis man skal teste et program, hvor udfaldet ikke altid vil være det samme. Denne gang tager jeg fat i eksemplet fra før, hvor man kunne indtaste sit navn og alder. Den opmærksomme læser vil nok have fundet ud af at for at gøre sådan, at programmer er sikret mindst et gennemløb, skal man bruge en do..while løkke.

medBrugerInput version 2:

```
//Opretter en tekststreng, der skal bruges i do...while løkken!
```

```
String fortsaet;

do {
    //Opretter en instans af klassen BufferReader, så vi kan modtage input fra
brugeren
    BufferedReader input = new BufferedReader(new
InputStreamReader(System.in));

    //Udskriver, hvad brugeren skal skrive af input
    //Og gemmer disse
    System.out.println("Skriv venligst dit fornavn:");
    String ditFornavn = input.readLine();

    System.out.println("Skriv venligst dit efternavn:");
    String ditEfternavn = input.readLine();

    System.out.println("Skriv venligst din alder");
    String dinAlder = input.readLine();


    //Udskriver
    System.out.println("Hej " + ditFornavn + " " + ditEfternavn + ".");
    System.out.println("Du er " + dinAlder + " aar gammel!");

    //Spørger om brugeren har lyst til at fortsætte programmet
    System.out.println("\n\n");
    System.out.println("Vil du fortsaette?");
    System.out.println("Tryk Y, hvis du vil.");
    System.out.println("Tryk alt andet, hvis nej");
    fortsaet = input.readLine();

    //Programmet fortsætter kun, hvis fortsaet = Y!
} while (fortsaet.equals("Y"));
```

Husk at oprette jeres "fortsaet" tekststreng **FØR** do..while løkken, da compileren ellers vil melde en fejl. Der er der en lille hage ved brug af **eqauls**. Hvis du skriver et lille y i stedet for et stort, vil programmet stoppe. Det er fordi, at eqauls skelner mellem store og små bogstaver og derfor vil y blive betragtet som et andet symbol end Y. For at løse dette skal du erstatte

```
} while (fortsaet.equals("Y"));
```

med

```
} while (fortsaet.equalsIgnoreCase("Y"));
```

Skriver man et lille y nu, vil programmet fortsætte.

Det var så min artikel, jeg håber, I kan bruge den til noget.

Milhøj

Kommentar af kasseper d. 02. Aug 2004 | 1

hmmm, sær exception handling....Meget sjovt koncept med de her artikler....

Kommentar af x-masman d. 31. Dec 2004 | 2

Ja. Det er vist meget for begyndere og sikkert fint til det. Det kunne måske være lidt mere spændende og kigge på parsning af input og udfra det lave en handling.

Kommentar af baitianlong d. 07. Aug 2004 | 3

rørende

Kommentar af jcpedersen d. 27. Jul 2004 | 4

Fint nok for en begynder...

Kommentar af krukken d. 23. Aug 2005 | 5

Hmm, ikke noget nyt under solen her.

Kommentar af duqe d. 26. Jul 2004 | 6

Kommentar af celron d. 09. Sep 2004 | 7

Kommentar af morteeart d. 13. Jan 2005 | 8

synes faktisk det er en god artikel, da jeg har stillet spøgsmål om lign.
Så slet er 5 point da heller ikke :P God til nybegyndere

Kommentar af schwarz84 d. 22. Jun 2005 | 9

Java bruges ikke ret meget til terminalprogrammer og det er nok derfor, det ikke er så let at kode noget der tager input i en terminal.
Undrer mig lidt over at du ikke behandler noget med at tage input med som argumenter til programmet først, men det er måske bevidst?
Exceptions er til for at blive fanget, ikke for at blive smidt i hovedet på brugeren. Om ikke andet, kan du jo lave en handler, som skriver en pæn streng ud som fejlmeddelelse.