Kursus 02199: Programmering
afsnit 3.1-3.5

Anne Haxthausen

IMM, DTU

1. Kontrol af programudførelsen (afsnit 3.1)
2. Valg-sætninger (**if** og **switch**) (afsnit 3.2 og 3.3)
3. Bloksætninger (afsnit 3.2)
4. Logiske udtryk (afsnit 3.2, 3.4)
5. Flere operatorer (læs selv afsnit 3.5)

Kontrol af programudførelsen

- Med mindre andet angives, udføres et program *lineært*:
  - Java udfører sætningerne i `main` metoden en efter en
- Udførelsen kan styres v.h.a. to specielle sætningstyper:
  - betingede sætninger (valg mellem en eller flere sætninger): **if-else** og **switch**.
  - løkker (gentagelse af sætninger): **while**, **do** og **for**.
- styringen foregår v.h.a. *logiske (Boolske) udtryk*:
  - ved betingelser angiver udtrykket hvilken gren, der skal udføres.
  - ved løkker angives *slutbetingelsen*.

`if`-sætninger

if-sætninger er god i situationer, hvor man kan sige:

"hvis ... så ..."

Eksempel:

```
double topskat = 0.0;

if (indkomst > 267000)
   topskat = (indkomst-267000) * 0.15;
```


if-else-sætninger

if-else-sætninger er god i situationer, hvor man kan sige:

"hvis ... så ... ellers ..."

Eksempel:

```
double topskat;

if (indkomst > 267000)
   topskat = (indkomst-267000) * 0.15;
else topskat = 0.0;
```

if-else-sætninger

Generelt format:

if (*udtryk*)
  sætning1
else
  sætning2

Virkning:

(1) beregn værdien af *udtryk*

(2) hvis **true** så udfør *sætning1*, ellers udfør *sætning2*

Blokke

En række sætninger kan grupperes til én sætning, en såkaldt *blok*, med `{ ... }`:

En blok kan forekomme hvor som helst en sætning kan forekomme.

Blokke er nyttige, når flere ting skal udføres på baggrund af et valg.

Eksempel:

```
if (indkomst > 267000)
   topskat = (indkomst - 267000) * 0.15;
else
{
  topskat = 0.0;
  System.out.println("Bedre␣held␣ved␣" +
                     "næste␣lønforhandling");
}
```


Blokke og if-else

Problem: else binder til sidste uafsluttede **if**.

```
if (x==3)
   if (y < 10) System.out.println("gren␣A");
else System.out.println("gren␣B");
```

Løsning:

```
if (x==3)
   { if (y < 10) System.out.println("gren␣A"); }
else
   {System.out.println("gren␣B");}
```

Ved brug af **if / if-else** inden i en anden **if-else** bør man bruge tuborg-klammer til at fremhæve sammenhængen.

Hellere for mange tuborg-klammer end logiske fejl!

International karaktergivning

Internationalt er man blevet enig om følgende karakterskala:

Score i %	Karakter
95–100	A
90–94	A-
87–89	B+
83–86	B
80–82	B-
67–69	D+
⋮	⋮
0–59	F

Opgave: Lav et program der omregner en procentscore til en karakter.

Karaktergivning med nestet if-else

```
int    score; // 0-100
String grade; // A, A-, ..., D-, F

if (score >= 95)
   grade = "A";
else
{
   if (score >= 90)
      grade = "A-";
   else
   {
      if (score >= 87)
         grade = "B+";
      else // o.s.v.
         ...
   }
}
```


switch-sætningen

switch er god, når der ønskes mere end to grene.

Karaktergivning med switch

```
int    score; // 0-100
String grade; // A, A-, ..., D-, F

switch (score)
{
    case 100: case 99: case 98: case 97: case 96:  case 95:
       grade = "A";
       break;
    case 94: case 93: case 92: case 91: case 90:
       grade = "A-";
       break;
    // o.s.v.
    default: // 0 - 59
       grade = "F";
}
```


Logiske udtryk

Vi har tidligere set eksempler på udtryk der angiver

- numeriske værdier (af type **int, double, ...**)
- tegnstrenge (af typen `String`)

Logiske udtryk angiver sandhedsværdier (af typen **boolean**). Logiske udtryk kan bl.a. være:

- **true** og **false**
- sammenligningsoperatorer påtrykt udtryk af samme type (f.eks. `indkomst > 267000`)
- logiske operatorer påtrykt logiske udtryk (f.eks. `!fundet && (x < 10)`)

Sammenligningsoperatorer og deres prædens

Operator	Betydning	Eksempel
<	Mindre end	`x < 60`
<=	Mindre end eller lig med	`x <= 60`
>	Større end	`x > 60`
>=	Større end eller lig med	`x >= 60`
==	Lig med	`x == 60`
!=	Forskellig fra	`x != 60`

Resultattypen er **boolean**, dvs. resultatet er **true** eller **false**.

De aritmetiske operatorer `*`, `/`, `%`, `+`, `-` binder alle stærkere end `<`, `<=`, `>`, `>=`.

Sammenligningsoperatorerne `<`, `<=`, `>`, `>=` binder alle stærkere end `==` og `!=`.

Eksempel: Hvad er resultatet af følgende logiske udtryk?

Lad `x` = 2 og `y` = 4.

Logisk udtryk	Resultatværdi
`false`	
`true`	
`true == false`	
`x != y`	
`x < 3 + y`	
`y < x + 3`	
`(x + y > 3) == false`	
`false != x < 3`	
`x == y == false`	

Sammenligning af tegn

To tegn kan – som tal – sammenlignes:

Eksempel:

```
char ch1= 'æ', ch2 = 'ø';

if (ch1 < ch2)
   System.out.println(ch1 +
                      "␣er␣mindre␣end␣" +
                      ch2);
else
   System.out.println(ch1 +
                      "␣er␣større␣end␣" +
                      ch2);
```

med følgende resultat ved kørsel:

```
æ er mindre end ø
```


Sammenligning af strenge

Strenge skal sammelignes v.h.a. metoderne `equals` og `compareTo`:

Eksempel:

```
String str1="første", str2="anden";

System.out.println("Er␣str1␣og␣str2␣ens?␣" +
                   str1.equals(str2));

if (str2.compareTo(str1) < 0)
    System.out.println("str1␣kommer␣efter␣str2");
else  if (str2.compareTo(str1) > 0)
      System.out.println("str2␣kommer␣efter␣str1");
```

giver

```
Er str1 og str2 ens? false
str1 kommer efter str2
```

Sammenligning af kommatal

To kommatal i en datamat er kun ens, hvis *alle* bits i deres interne repræsentation er ens.

Ofte er man i situationer, hvor tallene bare skal være tilstrækkeligt tæt på hinanden:

```
double x=3.00002, y=3.00001;
double v=3.02, w=3.01;
final double TOLERANCE = 1E-3;

if (Math.abs(x-y) < TOLERANCE)
   System.out.println("x og y er næsten ens");

if (Math.abs(v-w) > TOLERANCE)
   System.out.println("men det er v og w ikke!");
```

giver

```
x og y er næsten ens
men det er v og w ikke!
```


Logiske operatorer og deres præcedens

Operator	Betydning	Eksempel
!	Ikke (Negation)	`!(x == 60)`
&&	Og (Konjunktion)	`0 <= x && x < 60`
||	Eller (Disjunktion)	`x < 0 || x >= 60`

Argumenttyperne er **boolean** og resultattypen er **boolean**.

Operatoren ! binder stærkere end && som binder stærkere end ||.

! binder også stærkere end sammenligningsoperatorerne og de aritmetiske operatorer.

&& og || binder svagere end sammenligningsoperatorerne og de aritmetiske operatorer.

Sandhedstabeller for de logiske operatorer:

a	! a
false	true
true	false

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Udtrykket udregnes fra venstre mod højre.

Operatorerne && og || udregner ikke mere end højst nødvendigt:

Hvis *udtryk1* er falsk i *udtryk1* && *udtryk2* så udregnes *udtryk2* ikke.

Hvis *udtryk1* er sand i *udtryk1* || *udtryk2* så udregnes *udtryk2* ikke.

Eksempel: Hvad er resultatet af følgende logiske udtryk?

Lad x = 2 og y = 4.

Logisk udtryk	**Resultatværdi**
`! false`	
`! true`	
`! true == false`	
`! (true == false)`	
`true && false`	
`false || true`	
`x + y > 3 && x < y`	
`x + y == 3 || x < 4`	
`x < y && (3*4 == 2*6-1*2+2) == !(3<x)`	

Flere operatorer

Givet **int** `count = 0;`

```
count ++;
```

svarer til

```
count = count + 1;
```

Læs selv mere om dette i afsnit 3.5.

Råd: øg læsbarheden – skriv det fulde udtryk!

Husk til næste tirsdag (25/9)

- Læs afsnit 3.6-3.9.
- Forbered opgaverne.
- Afhent cd, hvis du har bestilt en.