UNIVERSITETET I OSLO

INF1000 høst 2010

Forelesning 3: Avrunding av elementær innføring

- Typer og typekonvertering
- Utregning av uttrykk
- String-klassen
- easyIO: Innlesning fra tastatur og fil
- easyIO: Skrive til fil
- Oppsummerende eksempel

UNIVERSITETET I OSLO

Java-program

- Variable **deklareres** og har en **type**
- Variable **tilordnes** verdier i setninger
- Setninger kan inneholde **uttrykk**
- Program:
  - skrives i en editor
  - lagres i .java-fil
  - kompileres til .class-fil
  - kjøres fra et kommandovindu
  - setninger eksekveres i sekvens, ovenfra nedover

UNIVERSITETET I OSLO

Numeriske typer

Type	Lovlige verdier
byte	-128 til 127
short	-32 768 til 32 767
int	-2^{31} til $2^{31}-1$
long	-2^{63} til $2^{63}-1$
float	-3.4e38 til 3.4e38
double	-1.7e308 til 1.7e308

UNIVERSITETET I OSLO

long og float

Verdiene krever spesialtegn for å instruere kompilatoren.

```
long verdi1 = 12345678901234;  // Feil: Tallet blir
// tatt som integer, men det er for stort for integer

long verdi2 = 12345678901234L; // OK, L betyr ”long”

float verdi3 = 12.345;  //Feil: dataverdien er double.
// Den kan ikke automatisk konverteres til float

float verdi4 = 12.345F; // OK, F betyr ”float”
```

UNIVERSITETET I OSLO

Typekonvertering

- Utvidelse av typen:
  - Tillater flere dataverdier
  - Automatisk
- Innsnevring av typen
  - Tillater færre dataverdier
  - Eksplisitt
- Rekkefølgen (utvidelse mot høyre)

```
byte, short, int, long, float, double
```

double => int

```
class Avrunding {
  public static void main (Stri
    double x = 0.53;

    System.out.print("Konvertering med (int): ");
    System.out.println( (int) x );

    System.out.print("Avrund nedover: ");
    System.out.println((int) Math.floor(x));

    System.out.print("Avrund oppover: ");
    System.out.println((int) Math.ceil(x));

    System.out.print("Avrund til nærmeste heltall : ");
    System.out.println((int) Math.round(x));
  }
}
```

```
$ javac   Avrunding.java
$ java   Avrunding
Konvertering med (int): 0
Avrund nedover: 0
Avrund oppover: 1
Avrund til nærmeste heltall : 1
$
```

UNIVERSITETET I OSLO

Inkrementering

- ++ øker verdien til variabelen med 1
- ++antall Endrer antall og returnerer den nye verdien
- antall++ Returnerer den gamle verdien og endrer deretter verdien
- NB! Dette er uttrykk som returnerer en verdi!

```
int svar1, svar2, i=0, j=0;
svar1 = ++i; // i blir først 1. Så returnerer uttrykket
             // ++i verdien 1, som svar settes lik
svar2 = j++; // uttrykket j++ returnerer 0, som svar
             // settes lik, så økes j til 1.
// Dermed: i==1, j==1, svar1==1, svar2==0
```

UNIVERSITETET I OSLO

Dekrementering

- --antall Minker antall med 1 og returnerer den nye verdien
- antall-- Returnerer den gamle verdien og minker deretter antall med 1

Side-effekter

- ++ og -- danner uttrykk der en variabel endres ved utregning:

```
svar = ++i; // endrer både verdien til svar og i
```

- Vi sier at slike uttrykk har *side-effekter*
- *Unngå uttrykk med side-effekter!*
  - men ++ og -- kan med fordel brukes i løkker på en kontrollert måte

Literaler

- Tallene vi skriver i programmet vårt kalles literaler.
- De som kun inneholder tall tolkes som **int**.

```
int verdi = 12345;
```

- De som inneholder desimaltegn tolkes som **double**.

```
double verdi2 = 12.345;
```

- Andre literaler er true/false og bokstaver:

```
boolean verdi3 = true;
char verdi4 = 'c';
```

Evaluering av uttrykk

Av og til kan vi unngå å sette parenteser. Eksempler:

2 + 5 * 4 - 3 **er lik** 2 + (5*4) – 3

b1 || b2 && b3 **er lik** b1 || (b2 && b3)

b1 && b2 == b3 || b4 **er lik**
(b1 && (b2 == b3)) || b4

Presedensregler

- **Presedensregler** angir hvilke operatorer som har fortrinn (førsterett) ved utregning av sammensatte uttrykk.
- F.eks. blir * beregnet før +. Vi sier at
  - * har **høyere** presedens enn +
  - + har **lavere** presedens enn *
- Reglene står i læreboka!
- Tips: Bruk parenteser!

Aritmetisk uttrykk – skjulte parenteser

```
2.0*3  + Math.ceil(7.23)    +  (2 + 3) * 3.5
2.0*3  + (Math.ceil(7.23))  +  (2 + 3) * 3.5
(2.0*3) + (Math.ceil(7.23))  +  (2 + 3) * 3.5
(2.0*3) + (Math.ceil(7.23))  + ((2 + 3) * 3.5)
((2.0*3) + (Math.ceil(7.23))) + ((2 + 3) * 3.5)
(((2.0*3) + (Math.ceil(7.23))) + ((2 + 3) * 3.5))
```

Aritmetisk uttrykk – utregning

```
(((2.0*3) + (Math.ceil(7.23))) + ((2 + 3) * 3.5))
((6.0    + (Math.ceil(7.23))) + ((2 + 3) * 3.5))
((6.0    +  8.0)              + ((2 + 3) * 3.5))
(14.0                         + ((2 + 3) * 3.5))
(14.0                         + ( 5      * 3.5))
(14.0                         +   17.5)
31.5
```

Eksempel – Logisk uttrykk

```
int u=1,v=1;
boolean b = true;

boolean resultat =
    u>2 || v<2 &&  b == !(++u == v++);

// Hva blir resultat?
```

Dette er et eksempel på et uttrykk med side-effekt!

Dere må aldri finne på å skrive noe slikt i et program!

Presedensregler for logiske uttrykk

- Høyest: Metodekall
- ! (ikke)
- <, <=, >, >=
- ==, !=
- && (og)
- || (eller)

Logisk uttrykk – skjulte parenteser

```
 u>2  ||   v<2  &&  b ==  !( ++u  ==  v++)

(u>2) ||  (v<2) &&  b ==  !((++u) == (v++))

(u>2) ||  (v<2) &&  b == (!((++u) == (v++)))

(u>2) ||  (v<2) && (b == (!((++u) == (v++))))

(u>2) || ((v<2) && (b == (!((++u) == (v++)))))

((u>2) || ((v<2) && (b == (!((++u) == (v++))))))
```

Logisk uttrykk – utregning

```
1:((u>2)  || ((v<2) && (b    == (!((++u) == (v++))))))
2:( false || ((v<2) && (b    == (!((++u) == (v++))))))
3:( false || (true  && (b    == (!((++u) == (v++))))))
4:( false || (true  && (true == (!((++u) == (v++))))))
5:( false || (true  && (true == (!(   2  == (v++))))))
6:( false || (true  && (true == (!(   2  ==   1  ))))))
7:( false || (true  && (true == (! false ))))
8:( false || (true  && (true == true )))
9:( false || (true  && true ))
10:(false || true)
11: true
```

- Start: u=1, v=1, b=true
- Utregning fra venstre mot høyre, beregnede verdier i rødt
- Når linjene 1-4 beregnes, er både u og v lik 1
- Når linje 7 beregnes, er både u og v lik 2

Eksempel – String

```
int x = 2, y = 3;

String tekst = "2+3==" + (x + y) + "!=" + x + y;

System.out.println("tekst: " + tekst);

// Hva blir skrevet ut?
```

String – skjulte parenteser

```
     "2+3==" + (x + y)  + "!="  + x + y

(    "2+3==" + (x + y)) + "!="  + x + y

((   "2+3==" + (x + y)) + "!=") + x  + y

(((  "2+3==" + (x + y)) + "!=”) + x) + y

((((“2+3==" + (x + y)) + "!=”) + x) + y)
```

String – utregning av verdi

```
((((”2+3==” + (x + y)) + ”!=”) + x) + y)

((((”2+3==” +  5     ) + ”!=”) + x) + y)

(((  ”2+3==5”          + ”!=”) + x) + y)

((   ”2+3==5!=”                + x) + y)

(    ”2+3==5!=2”                    + y)

     ”2+3==5!=23”
```

Tekster og klassen String

- En tekststreng er en sekvens av tegn (null, en eller flere), f.eks.
  - ””
  - ”&”
  - ”Arne er student”
- Hver tekststreng er et objekt av typen String
- String-objektet kan ikke endres (Immutable)
- For å finne lengden (antall tegn):

```
String s = ”kake”;
int lengde = s.length();
// lengde er nå 4
```


Bruk av spesialtegn

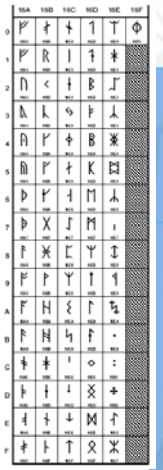
- Alle tegn kan angis som ’\uxxxx’ hvor hver x er en av

  0, 1, 2, ..., 9, A, B, C, D, E, F

  Eksempel: ’\u0041’ er tegnet ’A’

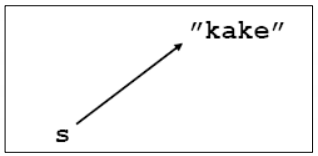
- Noen spesialtegn har egen forkortelse:
  - \t tabulator
  - \r carriage return (skriving starter først på linja)
  - \n linjeskift
  - \” dobbelt anførselstegn
  - \’ enkelt anførselstegn
  - \\ backslash

Unicode (http://www.unicode.org)



Objekter og pekere

- Vi lager *pekere* og *objekter* når vi bruker arrayer og strenger
  - `int [] a = new int [1024];`
  - `String s = "kake";`
- Selve variabelen er en peker som:
  - Inneholder adressen til hvor objektet er i hukommelsen
  - Tegnes som en pil



null

- Hva om vi *ikke* har laget et objekt ?
  - `int [] b ;`
  - `String navn;`
- Hva peker de på? Svar: *ingenting*!
- Denne 'ingenting'-verdien heter **`null`**

`null` kan testes på

```
if (navn == null)
  System.out.print("navn mangler");
if (b != null)
  System.out.print("arrayen b finnes");
if (b != null && b[0] > 10)
  System.out.print("b[0] er stor nok");
else Syste.out.print ("enten finnes ikke
   arrayen b eller b[0] er for liten");
navn = null;
```

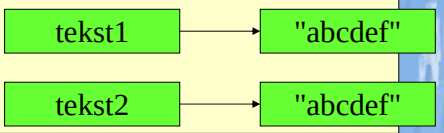
equals() tester om to tekster er like

Anta at s og t er tekstvariable (og s ikke er **null**)

```
if (s.equals(t)) {
    System.out.println("Tekstene er like");
} else {
    System.out.println("Teksten er forskjellige");
}
```

Bruk av == virker av og til, men ikke alltid:

```
String s = "abc";
String t = "def";
String tekst1 = s + t;
String tekst2 = s + t;
```



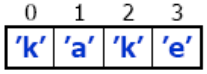
`tekst1.equals(tekst2)`er true

`tekst1 == tekst2` er **false** siden pekerne er forkjellige

UNIVERSITETET I OSLO

De enkelte tegnene i en tekststreng

Tegnene i en tekststreng har posisjoner indeksert fra 0 og oppover



Vi kan få tak i tegnet i en bestemt posisjon:

```
String s = ”kake”;
char c = s.charAt(1);
// Nå er c == ’a’
```

Vi kan erstatte alle forekomster av et tegn med et annet tegn:

```
String s1 = ”kake”;
String s2 = s1.replace(’k’, ’r’);
// Nå er s2 en referanse til tekststrengen ”rare”
```



```
class ReplaceOppgave {
    public static void main (String []  args) {
        String s = "javaprogram";
        String l = s;
        s.replace('a', 'i');
        l.replace('a', 'o');
        System.out.println(s);
        System.out.println(l);

        l="jp";
        System.out.println(s);
    }
}
```

Husk: replace legger resultatet i en ny string!

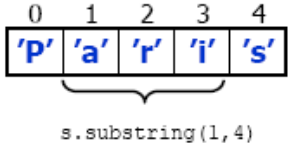
```
$ javac ReplaceOppgave.java
$ java ReplaceOppgave
javaprogram
javaprogram
javaprogram
$
```

UNIVERSITETET I OSLO

Deler av en tekststreng

Vi kan trekke ut en del av en tekststreng:

```
String s = ”Paris”;
String s1 = s.substring(1,4);
// Nå er s1 tekststrengen ”ari”
```



Generelt:

```
s.substring(index1, index2)
```

Første posisjon som skal være med

Første posisjon som ***ikke*** skal være med

Siste del av en tekststreng:

```
String s = ”Paris er hovedstaden i Frankrike”;
String s1 = s.substring(6);
// Nå er s1 tekststrengen ”er hovedstaden i Frankrike”
```


UNIVERSITETET I OSLO

Alfabetisk ordning

- Anta at s og t er tekstvariable (og at s ikke har verdien null)
- Er s foran t i alfabetet?

```
int k = s.compareTo(t);
if (k < 0) {
    System.out.println("s er alfabetisk foran t");
} else if (k == 0) {
    System.out.println("s og t er like");
} else {
    System.out.println("s er alfabetisk bak t");
}
```

Husk at det er Unicode-verdien som brukes her og at det kan gi uventet resultat!

```
import easyIO.*;
class Alfabetisk {
  public static void main (String []  args) {
      String sString = "abCDøÅ";
      String tString = "bCdDØÆ";

      for(int i=0; i < sString.length();i++){
        String s = sString.substring(i, i+1);
        String t = tString.substring(i, i+1);

        int k = s.compareTo(t);

        if (k < 0) {
            System.out.print(s + " er alfabetisk foran " + t);
        } else if (k == 0) {
            System.out.print(s + " og " + t + " er like");
        } else {
            System.out.print(s + " er alfabetisk bak " + t);
        }
        System.out.println("\t\t k er " + k);
    }
  }
}
```

```
M:\INF1000\Programmer>java Alfabetisk
a er alfabetisk foran b          k er -1
b er alfabetisk bak C            k er 31
C er alfabetisk foran d          k er -33
D og D er like          k er 0
° er alfabetisk bak ╪            k er 32
┼ er alfabetisk foran ╞          k er -1
```

```
arnem@sviurr ~/INF1000/Programmer> java Alfabetisk
a er alfabetisk foran b          k er -1
b er alfabetisk bak C            k er 31
C er alfabetisk foran d          k er -33
D og D er like          k er 0
ø er alfabetisk bak Ø            k er 32
Å er alfabetisk foran Æ          k er -1
```

UNIVERSITETET I OSLO

Inneholder en tekst en annen?

```
String s = ”javaprogram”;
String t = ”program”;

int k = s.indexOf(t);

if (k < 0)
    System.out.println("s inneholder ikke t");
else {
    System.out.println("s inneholder t");
    System.out.println("Posisjon i s: " + k);
}
```

```
s inneholder t
Posisjon i s: 4
```

UNIVERSITETET I OSLO

Starter en tekst med en annen?

- Anta at s og t er tekstvariable (og at s ikke har verdien null)
- Starter s med teksten t?

```
boolean b = s.startsWith(t);
if (b) {
    System.out.println("s starter med t");
} else {
    System.out.println("s starter ikke med t");
}
```

```
import easyIO.*;
class StarterMedOppgave {
    public static void main (String []  args) {
        String s = "julenisse";
        String t = s.substring(4);
        String v = s.substring(0,4);
        if(s.startsWith("jule")){
            System.out.println("A");
        }
        if(t.startsWith("jule")){
            System.out.println("B");
        }
        if(v.startsWith("jule")){
            System.out.println("C");
        }
    }
}
```

```
$ javac StarterMedOppgave.java
$ java StarterMedOppgave
A
C
$
```

UNIVERSITETET I OSLO

Slutter en tekst med en annen?

- Anta at s og t er tekstvariable (og at s ikke har verdien null)
- Slutter s med teksten t?

```
boolean b = s.endsWith(t);
if (b) {
    System.out.println("s ender med t");
} else {
    System.out.println("s ender ikke med t");
}
```


UNIVERSITETET I OSLO

Fra tall til tekst og omvendt

For å konvertere fra tall til tekst:

```
String s1 = String.valueOf(3.14);
String s2 = String.valueOf('a');
String s3 = String.valueOf(false);
String s4 = "" + 3.14;
String s5 = "" + 'a';
String s6 = "" + false;
```

For å konvertere fra tekst til tall:

```
int k = Integer.parseInt(s);
double x = Double.parseDouble(s);
```

og tilsvarende for de andre numeriske datatypene…

UNIVERSITETET I OSLO

Lesemetoder i easyIO

```
// Opprette forbindelse med tastatur:
In tastatur = new In();

// Lese et heltall:
int k = tastatur.inInt();

// Lese et desimaltall:
double x = tastatur.inDouble();

// Lese et enkelt tegn:
char c = tastatur.inChar();

// Lese et enkelt ord:
String s = tastatur.inWord();

// Lese resten av linjen:
String s = tastatur.inLine();
```

Hvilken lesemetode skal jeg velge?

- Først:
  - importere easyIO og åpne forbindelse til tastaturet.
- Lese item for item:
  - For å lese et heltall: inInt() [egentlig: tastatur.inInt()]
  - For å lese et desimaltall: inDouble()
  - For å lese ett ord: inWord()
  - For å lese en hel linje (med minst ett tegn): inLine()
- Lese linje for linje:
  - Bruk readLine()
- Lese tegn for tegn:
  - For å lese neste tegn (også hvite tegn): inChar()

Hvordan lesemetodene virker

- **inInt()**, **inDouble()** og **inWord()**:
  - Hopper over eventuelle innledende blanke tegn.
  - Leser så alt fram til neste blanke tegn eller linjeskift
  - Hvis innlest data har feil type, gis en feilmelding og man får en ny sjanse (3 sjanser)
- **inChar()**:
  - Leser neste tegn, enten det er et blankt tegn eller ikke
- **inLine()**:
  - Leser alt fram til slutten av linjen (inkludert blanke tegn)
  - Ignorerer linjer hvor det kun står (igjen) et linjeskift

Formatert utskrift til skjerm

- Formatert utskrift vil si at vi angir nøyaktig hvordan utskriften skal se ut og plasseres på skjermen.
  - Kan gjøres "manuelt" med System.out.print(...), men det er upraktisk.
- Bedre: bruke en ferdiglaget pakke for slikt. I INF1000 bruker vi pakken easyIO. I toppen av programmet (**før** class) skriv:

```
import easyIO.*;
```

- Inne i klassen skriver vi så:

```
Out skjerm = new Out();
```

- Så kan vi skrive ut det vi ønsker, f.eks.:

```
double pi = 3.1415926;
skjerm.out(pi, 2, 6); // Skriv ut pi med 2 desimaler
                      // høyrejustert på 6 plasser.
```

Eksempel

```
import easyIO.*;
class FormatertUtskrift {
  public static void main (String [] args) {
      Out skjerm = new Out();

      int BREDDE1 = 20;
      int BREDDE2 = 30;

      skjerm.out("NAVN", BREDDE1);
      skjerm.outln("ADRESSE", BREDDE2);
      skjerm.out("Kristin Olsen", BREDDE1);
      skjerm.outln("Vassfaret 14, 0773 Oslo",
            BREDDE2);
  }
}
```

Vi må først importere pakken easyIO.

Vi oppretter en verktøykasse for skriving til terminal

I verktøykassen ligger det bl.a. verktøy (på java-språk: *metoder*) for å skrive til skjerm med og uten linjeskift til slutt.

Dukket objekter opp her?

Resultat

```
$ javac FormatertUtskrift.java
$ java FormatertUtskrift
NAVN                ADRESSE
Kristin Olsen       Vassfaret 14, 0773 Oslo
```

Tre måter å skrive ut på

- Uten formatering:

```
skjerm.out("Per Hansen");
skjerm.out(12345);
skjerm.out(3.1415, 2);
```

- Angi utskriftsbredde:

```
skjerm.out("Per Hansen", 15); // Bredde 15 tegn
skjerm.out(12345, 15);        // Bredde 15 tegn
skjerm.out(3.1415, 2, 15);    // Bredde 15 tegn,
                              // 2 desimaler
```

- Angi utskriftsbredde og justering:

```
skjerm.out("Per Hansen", 15, Out.RIGHT); // Høyrejuster
skjerm.out(12345, 15, Out.CENTER);       // Senterjuster
skjerm.out(3.1415, 2, 15, Out.LEFT);     // Venstrejuster
```

Lese og skrive fra/til fil

- Klassene In og Out i easyIO
  - Les dokumentasjonen!
- In og Out + Format
  - brukes i INF1000
  - Format brukes til mer 'finjustert' formattering
  - Det er flere metoder enn de som gjennomgås
- easyIO ble laget fordi Javas innebygde IO-metoder var for kompliserte
  - Java bedre nå (Java skanner)
  - men fortsatt noe vanskeligere enn easyIO

Eksempel

Vi importerer pakken easyIO.

Vi åpner filen for lesing

Her leses hele første linje av filen

```
import easyIO.*;

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

        In fil = new In("filnavn");

        String s = fil.inLine();

        System.out.println("Første linje var: "
             + s);
    }
}
```

Lese ord for ord (item)

UNIVERSITETET I OSLO

- Metoder:
  - inInt() for å lese et heltall
  - inDouble() for å lese et flyttall
  - inWord() for å lese et ord
  - lastItem() for å sjekke om slutten av filen er nådd
- Eksempel: lese en fil tall for tall

```
In fil = new In(”item.txt");
while (!fil.lastItem()) {
    int k = fil.inInt();
    System.out.println(”Tallet var ” + k);
}
```


Lese ord for ord: skilletegn

UNIVERSITETET I OSLO

Hopper over skilletegn mellom ordene man leser:

- linjeskift-tegnene (+ noen sære tegn) er alltid skilletegn
- Hvis man ikke gjør noe er også blanke, tab,... skilletegn
- Brukeren kan også spesifisere skilletegn:
  - String egneSkilletegn = ”F(,)”;
  - i = inInt(egneSkilletegn);
  - ***før*** det neste ord leses ignoreres nå kun linjeskift, samt tegnene i ’egneSkilletegn’

Lese tegn for tegn

UNIVERSITETET I OSLO

- inChar(false) returnerer det første uleste tegnet (uansett om det er skilletegn eller ikke)
  - Kan kopiere en fil tegn for tegn uansett hva den inneholder
- inChar(true) returnerer det første uleste tegnet som ikke er skilletegn
- Ser etter slutten med **endOfFile()**

Lese linje for linje

UNIVERSITETET I OSLO

- Metoder:
  - readLine() for å lese en linje
  - inLine() for å lese resten av en linje (leser neste linje hvis det ikke er mer igjen enn linjeskift på nåværende linje)
  - endOfFile() for å sjekke om slutten av filen er nådd
- Eksempel: lese en fil linjevis

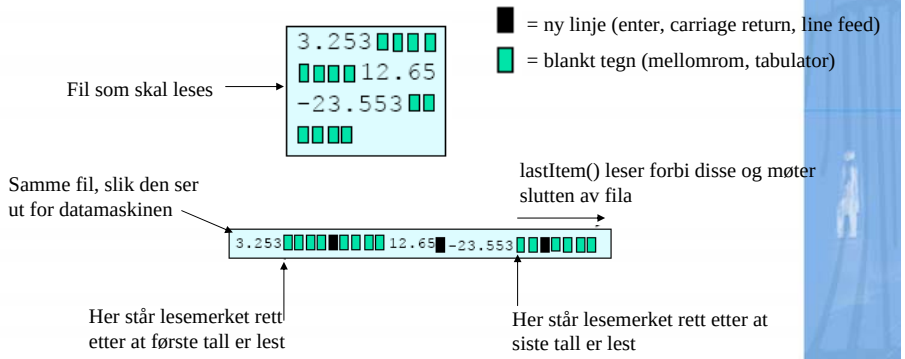
```
In fil = new In("fil.txt");
while (!fil.endOfFile()) {
    String s = fil.readLine();
    System.out.println(”Linjen var ” + s);
}
```

```
import easyIO.*;
class LinjeForLinje {

    public static void main (String[] args) {
        In innfil = new In(”filnavn");
        String[] s = new String[100];
        int ant = 0;
        while (!innfil.endOfFile()) {
            s[ant] = innfil.readLine();
            ant = ant + 1;
        }
        for (int i=0; i<ant; i++) {
            System.out.println(s[i]);
        }
    }
}
```


lastItem og endOfFile, lesemerket

- **endOfFile()** sjekker kun om siste tegn på fila er lest
- **lastItem()** søker seg fram til første ikke-blanke tegn og returnerer **true** hvis slutten av fila nås og **false** ellers.



Når er vi ferdige med å lese en fil?

- Vi må ha mulighet til å avgjøre når slutten av filen nådd
  - ellers kan det oppstå en feilsituasjon
  - Metodene lastItem() og endOfFile() kan benyttes til dette
- Noen ganger er filens lengde kjent på forhånd:
  - lengden er kjent før programmet kjøres
  - lengden ligger lagret i begynnelsen av filen
- Da kan vi i stedet benytte en for-løkke.

Eksempel: fil med kjent lengde

- Lag program som leser en fil med
  - 10 desimaltall
  - tallene er atskilt med blanke tegn og/eller linjeskift
  - lagrer tallene i en array
- Algoritme:
  - For-løkke med 10 gjennomløp
  - Bruk inDouble-metoden fra easyIO

```
import easyIO.*;
class Les10Tall {
    public static void main (String[] args) {
        double[] x = new double[10];
        In innfil = new In(”tall.txt");
        for (int i=0; i<10; i++) {
            x[i] = innfil.inDouble();
        }
        // Nå kan vi evt. gjøre noe med
        // verdiene i arrayen x
    }
}
```

Nok at tallene er atskilt

UNIVERSITETET I OSLO

Programmet på forrige foil ville gitt akkurat samme resultat for alle disse filene:

```
15.2
6.23
3.522
3.6
8.893
-3.533
65.23
22.01
45.02
7.2
```

```
15.2  6.23
3.522 3.6
8.893 -3.533
65.23 22.01
45.02 7.2
```

```
15.2   6.23   3.522  3.6
8.893  -3.533
65.23  22.01  45.02


       7.2
```


UNIVERSITETET I OSLO

Eksempel: fil med lengde-info

- Lag program som leser en fil med
  - på forhånd ukjent antall desimaltall
  - tallene er atskilt med blanke tegn og/eller linjeskift
  - lagrer tallene i en array
  - *antall tall som skal leses ligger øverst i filen*
- Algoritme:
  - Les antall tall fra fila
  - For-løkke med så mange gjennomløp som det er tall
  - Bruk inDouble-metoden fra easyIO


```
import easyIO.*;
class LesTallMedLengde {
  public static void main (String[] args) {
    // vet ikke lengden ennå double[] x;
    In innfil = new In(“tall-med-lengde.txt”);
    int lengde = innfil.inInt();
   // nå vet vi lengden
    x = new double[lengde];
    for (int i=0; i<lengde; i++) {
        x[i] = innfil.inDouble();
    }
    for (int i=0; i<lengde; i++) {
       System.out.println( i + ” = ” + x[i]);
    }
  }
}
```

Eksempel: fil med sluttmerke

- Lag program som leser en fil med
  - på forhånd ukjent antall desimaltall
  - tallene er atskilt med blanke tegn og/eller linjeskift
  - lagrer tallene i en array
  - *slutten av filen er markert med tallet -999*
  - *antall tall er max 100*
- Algoritme:
  - while-løkke inntil -999 leses
  - Bruk inDouble-metoden fra easyIO


```
import easyIO.*;
class LesTallMedMerke {
  public static void main (String [] args) {
    // antar max 100 tall på fil
    double [] x = new double[100];
    In innfil = new In(”tall-med-merke.txt");
    double siste = 0;
    int ant = 0;
    while (siste != -999) {
      siste = innfil.inDouble();
      if (siste != -999) {
        x[ant] = siste;
        ant = ant + 1;
      }
    }    // Nå ligger det verdier i
  }      // x[0], x[1], ...., x[ant-1]
}
```

Mer komplisert filformat

Anta at vi skal lese en fil med følgende format:

- Først en linje med 3 overskrifter
- Deretter en eller flere linjer på formen:
  - heltall, desimaltall, tekststreng
- Alle felt er separert av blanke tegn

```
Antall  Pris   Varenavn
35      23.50  Oppvaskkost
53      33.00  Kaffe
97      27.50  Pizza
...     ...    ...
...     ...    ...
```


Fremgangsmåte

- Den første linja er spesiell
  - vi tenker oss her at den ikke er så interessant
  - vi leser forbi den med readLine() eller inLine()
- De andre linjene har samme format,
  - løkke hvor hvert gjennomløp av løkken leser de tre itemene
  - bruker da henholdsvis inInt(), inDouble() og inWord().
- For å vite når filen er slutt:
  - kan enten bruke endOfFile() eller lastItem()
  - vi leser filen itemvis, bruker defor lastItem()
  - da får vi ikke problemer med blanke helt på slutten av filen
  - ofte et siste linjeskift på siste linje!

```
import easyIO.*;
class LesVarer {
    public static void main (String[] args) {
        In innfil = new In(”varer.txt");
        int []   t = new int[100];
        double[] x = new double[100];
        String[] s = new String[100];
        int ant = 0;
        innfil.readLine();
        while (!innfil.lastItem()) {
            t[ant] = innfil.inInt();
            x[ant] = innfil.inDouble();
            s[ant] = innfil.inWord(”\n”);
            ant = ant + 1;
        }
        for (int i=0; i<ant; i++) {
            System.out.println(t[i] + ”: ” + x[i] +
                  ” - ” + s[i]);
        }
    }
}
```

```
35: 23.5 - Oppvaskkost
53: 33.0 - Kaffe
97: 27.5 - Pizza
```

```
import easyIO.*;
class Skilletegn {
    public static void main (String[] args) {
        int r,k;
        String skille =" S(,)";
        In tast = new In(); Out skjerm = new Out();

        skjerm.out("Gi rad r og kollonne k som S(r,k): ");
        r = tast.inInt(skille);
        k = tast.inInt(skille);

        skjerm.outln("Du ga r=" +r+", og k=" +k);
    }
}
```

```
$ javac Skilletegn.java
$ java Skilletegn
Gi rad r og kollonne k som S(r,k): S(2,2)
Du ga r=2, og k=2
$
```

UNIVERSITETET I OSLO

Noen nyttige hjelpemidler (ikke pensum)

- Sjekke om det finnes en fil med et bestemt navn:

```
if (new File("filnavn").exists())
   System.out.println("Filen finnes");
```

- Slette en fil:

```
if (new File("filnavn").delete())
    System.out.println("Filen ble slettet");
```

- Avgjøre hvilket filområde programmet ble startet fra:

```
String curDir = System.getProperty("user.dir");
```

- Lage liste over alle filer og kataloger på et filområde:

```
String [] allefiler = new File(filområdenavn).list();
```

Skrive til fil

```
import easyIO.*;
class SkrivTilFil {
    public static void main (String[]  args) {

        Out fil = new Out(”nyfilnavn");

        fil.outln("Dette er første linje");

        fil.close();
    }
}
```

Vi importerer pakken easyIO.

Vi åpner filen for Skriving

Her skrives en linje med tekst til filen

Vi må huske å lukke filen til slutt

EasyIO: Skrivemetoder

Datatype	Eksempel	Beskrivelse
int	fil.out(x); fil.out(x, 6);	Skriv x Skriv x høyrejustert på 6 plasser
double	fil.out(x, 2); fil.out(x, 2, 6);	Skriv x med 2 desimaler Skriv x med 2 desimaler på 6 plasser
char	fil.out(c);	Skriv c
String	fil.out(s); fil.out(s, 6);	Skriv s Skriv s på 6 plasser (venstrejustert)
	fil.outln();	Skriv en linjeskift
	fil.close();	Lukk filen

Merk: dersom antall plasser spesifiseres og det ikke er nok plass, vil det som skrives ut avsluttes med tre punktumer: ...

Alternativ til if-else: switch

En sammensetning av flere if-setninger kan i noen tilfeller erstattes med en switch-setning:

```
switch (uttrykk) {
    case verdi1:
        <instruksjoner>
        break;
    ....
    case verdiN:
      <instruksjoner>
        break;
    default:
      <instruksjoner>
}
```

<uttrykk> må være **int**, **byte**, **short** eller **char**

Verdiene må være **konstanter** (ikke variable eller uttrykk). Samme verdi flere ganger gir kompileringsfeil

Eksekvering av switch-setning

- Uttrykkene regnes først ut
- Det letes etter et treff ovenfra og nedover.
- Utførelsen fortsetter ved den verdien som matcher
- Ved treff utføres setningene frem til ***break;***. Etter break er man ferdig med switch-setningen
  - Vær obs på at dersom det ikke står *break* vil utførelsen fortsette gjennom neste *case*
  - Bruk alltid *break !*
- Dersom det ikke finnes noen match vil setningene etter **default** utføres


```
class BrukAvSwitch {
  public static void main (String [] args) {
     char c;
     c = // en innlest verdi
     switch(c) {
     case 'a':
          System.out.println("Tegnet var en a");
          break;
     case 'b':
          System.out.println("Tegnet var en b");
          break;
     default :
          System.out.println(
                   "Tegnet var ikke a eller b");
     }
  }
}
```

UNIVERSITETET I OSLO

Oppsummerende eksempel

- Input: Fil fra kurssiden med forelesingsplanen
  - Hver linje har en dato og navnet på foreleser
  - Forelesningsnr. beregnes under innlesning
- Programmet har meny med valg
  - 1: Vis forelesningsplan.
  - 2: Endre foreleser fra nr.
  - 3: Vis nr fra foreleser.
  - 4: Lagre.
  - 5: Avslutt.
  - 0: Meny.

forelesningsplan.txt

UNIVERSITETET I OSLO

```
24.08.2010      Arild Waaler
31.08.2010      Arild Waaler
07.09.2010      Arild Waaler
14.09.2010      Ragnhild Kobro Run
21.09.2010      Ragnhild Kobro Runde
28.09.2010      Ragnhild Kobro Runde
05.10.2010      Arild Waaler
12.10.2010      Arild Waaler
19.10.2010      Arild Waaler
26.10.2010      Arild Waaler
02.11.2010      Arne Maus
09.11.2010      Ragnhild Kobro Runde
16.11.2010      Ragnhild Kobro Runde
23.11.2010      Arild Waaler
```

Flere forskjellige skilletegn når data hentes fra kurssiden!

```
import easyIO.*;
class VisForelesningsplan {
    public static void main (String [] args) {
        // Les inn filen og lagre i datastruktur
        int valg = 0;
        while( valg!=5 ){
            switch( valg ){
                case 0: // Skriv meny til skjerm
                    break;
                case 1: // Skriv ut hele forelesningsplanen
                    break;
                case 2: // Les inn nr foreleser og endre data
                    break;
                case 3: // Les inn foreleser og list alle forelesninger
                    break;
                case 4: // Skriv tilbake til filen
                    break;
                default:
                    System.out.println("Ulovlig valg");
            }
            System.out.println();
            System.out.print("Skriv valg: ");
            valg = tastatur.inInt();
        }
    }
}
```

case 0: Skriv meny

UNIVERSITETET I OSLO

```
Out skjerm = new Out();

String meny = "1: Vis forelesningsplan. 2: Endre
 foreleser fra nr.  3: Vis nr fra foreleser.  4:
 Lagre.  5: Avslutt.  0: Meny.";

switch( valg ){
 case 0: skjerm.outln(meny); break; ...
}
```

```
$ javac VisForelesningsplan.java
$ java VisForelesningsplan
1: Vis forelesningsplan. 2: Endre foreleser fra nr.  3:
  Vis nr fra foreleser.  4: Lagre.  5: Avslutt.  0: Meny.

Skriv valg: _
```

Les inn fil og lagre data

```
In innFil = new In("forelesningsplan.txt");

String[] dato = new String[100];
String[] foreleser = new String[100];
int antall = 0;

while (!innFil.lastItem ()){
     dato[antall] = innFil.inWord();
 foreleser[antall] = innFil.inLine().trim();
     antall++;
}
```

Variabelen antall tjener to hensikter:
- Lagre antall registrerte forelesninger
- Være èn større enn indeksen til sist lagrede

trim fjerner skilletegn før og etter ord

case 1: Vis forelesningsplan

```
switch( valg ){
...
  case 1:
    for(int i=0; i<antall; i++){
       skjerm.out(i+1); skjerm.out(" ");
       skjerm.out(dato[i]); skjerm.out(" ");
       skjerm.outln(foreleser[i]);
    }
    break;

 ...
 }
```

```
Skriv valg: 1
1 24.08.2010 Arild Waaler
2 31.08.2010 Arild Waaler
3 07.09.2010 Arild Waaler
4 14.09.2010 Ragnhild Kobro Runde
5 21.09.2010 Ragnhild Kobro Runde
6 28.09.2010 Ragnhild Kobro Runde
7 05.10.2010 Arild Waaler
8 12.10.2010 Arild Waaler
9 19.10.2010 Arild Waaler
10 26.10.2010 Arild Waaler
11 02.11.2010 Arne Maus
12 09.11.2010 Ragnhild Kobro Runde
13 16.11.2010 Ragnhild Kobro Runde
14 23.11.2010 Arild Waaler
```

case 2: Endre foreleser fra nr

```
In tastatur = new In();

switch( valg ){
  case 2:
    skjerm.out("Nr: ");
    int indeks = tastatur.inInt()-1;
    skjerm.out("Skriv foreleser: ");
    foreleser[indeks] = tastatur.inLine();
    break;

 ...
}
```

Husk at forelesning nr i ligger på indeks i-1!

case 3: Vis nr fra foreleser

```
switch( valg ){
  case 3:
    skjerm.out("Foreleser: ");
    String innlestNavn = tastatur.inLine();
    for(int i=0; i<antall; i++)
      if( foreleser[i].equals(innlestNavn) ){
          skjerm.out(i+1); skjerm.out(" ");
          skjerm.out(dato[i]); skjerm.out(" ");
          skjerm.outln(foreleser[i]);
      }
    break;
 ...
 }
```

case 4: Lagre

```
switch( valg ){
 case 4:
    Out utFil = new Out("forelesningsplan.txt");
    for(int i=0; i<antall; i++){
      utFil.out(dato[i]); utFil.out(" ");
      utFil.outln(foreleser[i]);
    }
    utFil.close();
    skjerm.outln("Endringer er lagret");
    break;
...
}
```

```
Skriv valg: 3
Foreleser: Arild Waaler
1 24.08.2010 Arild Waaler
2 31.08.2010 Arild Waaler
3 07.09.2010 Arild Waaler
7 05.10.2010 Arild Waaler
8 12.10.2010 Arild Waaler
9 19.10.2010 Arild Waaler
10 26.10.2010 Arild Waaler
14 23.11.2010 Arild Waaler

Skriv valg: 2
Nr: 7
Skriv foreleser: Ragnhild Kobro Runde

Skriv valg: 3
Foreleser: Ragnhild Kobro Runde
4 14.09.2010 Ragnhild Kobro Runde
5 21.09.2010 Ragnhild Kobro Runde
6 28.09.2010 Ragnhild Kobro Runde
7 05.10.2010 Ragnhild Kobro Runde
12 09.11.2010 Ragnhild Kobro Runde
13 16.11.2010 Ragnhild Kobro Runde

Skriv valg: 4
Endringer er lagret
```