UNIVERSITETET I OSLO

INF1000 – Uke 3

Innlesing fra terminal, formatert utskrift og forgreininger

UNIVERSITETET I OSLO

Oversikt

- Underveisevaluering
- Innlevering av obligatorisk oppgave
- Litt repetisjon
  - Program
  - Variabler og Uttrykk
  - Presedens
  - Matematiske funksjoner
- Innlesing
- Formattert utskrift

UNIVERSITETET I OSLO

Spørsmål

- Hvor mange har kjørt et program?
- Hvor mange fikk programmet til å gjøre det de ville?
- Hvor mange har gjort ukeoppgavene?
- Hvor mange har gjort den obligatoriske oppgaven, men ikke nødvendigvis levert?

- Jo mer programmering dess bedre!
- Men, dere kan ikke bruke maskinen på eksamen, så øv på papir også.

UNIVERSITETET I OSLO

Underveisevaluering

EVALUERING AV EMNER:

Institutt for informatikk ønsker en kontinuerlig evaluering av både form og innhold i undervisningen. Evalueringen skal gi studentene ved et emne mulighet til å komme med tilbakemeldinger underveis, slik at eventuelle forbedringer kan gjøres umiddelbart. I tillegg skal underveisevalueringen hjelpe faglærer og instituttet til å fange opp god og mindre god undervisningspraksis og heve kvaliteten på emnet/undervisningen.

Emneansvarlig lærer utformer evalueringsopplegget i samråd med studentene som følger emnet og er ansvarlig for kunngjøring av tidspunkt og gjennomføring. Omfang og evalueringsmetode tilpasses hvert enkelt emne og avgjøres av faglærer. Faglærer utfører eventuelle forbedringer og kommuniserer resultatet til studentene.

Du finner mer informasjon om dette på:
www.ifi.uio.no/studinf/kvalitetssikring/studenter

Med ønske om et godt semester
Institutt for informatikk

I praksis

- Ta opp saker med gruppelærerne
- Jeg snakker med de hver uke og vil gjerne vite hva som kan forbedres
- Etter noen forelesninger til holder vi en undersøkelse ved hjelp av ”flervalgsverktøyet”
- Kurskritikk etter at emnet er ferdig arrangeres av Fagutvalget for informatikk (FUI) http://heim.ifi.uio.no/~fui

PING [simula . research laboratory] ubuntu NORGE OLUG

Gratis hjelp til å installere Linux

Free Linux installation help

Tirsdag 4. september klokken 17.00
Tuesday the 4th of September at 17:00

Program-, Informasjons- og Nettverksteknologisk Gruppe ved Universitetet i Oslo (PING), Oslo Linux User Group (OLUG), Simula Research Laboratory og Ubuntu Norge har gleden av å invitere alle til installasjonskveld, hvor du får hjelp til å installere GNU/Linux på din bærbare eller stasjonære PC. Adresse: Preklinisk Odontologi (bygning nr. 16 på kartet), rom 125, vis à vis biblioteket.

Det er viktig at du har tatt sikkerhetskopi av viktige data før du kommer!

PING, OLUG, Simula Research Laboratory and Ubuntu Norway invites all to a GNU/Linux installation gathering, in PING's meeting room at Preklinisk Odontologi (building no. 16 on the map), room 125, opposite the library.

Please backup important files before the meeting!

http://ping.uio.no/
https://olug.no/
http://www.simula.no/
http://www.ubuntu.no/

Obligatorisk oppgave

- Frist: Fredag 7/9 kl. 23:59:59
- Skal leveres i systemet Joly.
  - **Vent til i morgen!**
- Dersom Joly ikke virker kan du levere på papir til gruppelærer på den gruppen du er registrert **første** gruppetime etter fristen.

- Husk å levere **noe** selv om det ikke virker helt som det skal!
- Alle skal levere unik besvarelse. Den må **ikke være for lik** en annens besvarelse (eller en fra tidligere år). Det er ikke lett å gjøre oblig 1 annerledes.

Fordeler med elektronisk innlevering – Joly

- For studentene
  - Du får en kvittering på når du har levert hver oblig
  - Har du en slik kvittering, så kan ingen si at du ikke har levert
  - Obligen din blir lagret permanent i en database – går ikke tapt
- For INF1000, gruppelærerne, og IfI
  - Vi får bedre orden på innlevering av obliger (sikring av innlevingene)
  - Vi får sjekket om noen innleverer kopier av andres besvarelser (uten Joly gjør minst 8-10% det)
  - Antall kopibesvarelser synker sterkt, studentene gjør mer selv
- Joly er skrevet av 4 master-studenter:
  - Chr. K.Kielland, Hanne Vibekk, Theresese Stensen og Cato Morholt (slike systemer kan kanskje også du lage om 3 år)

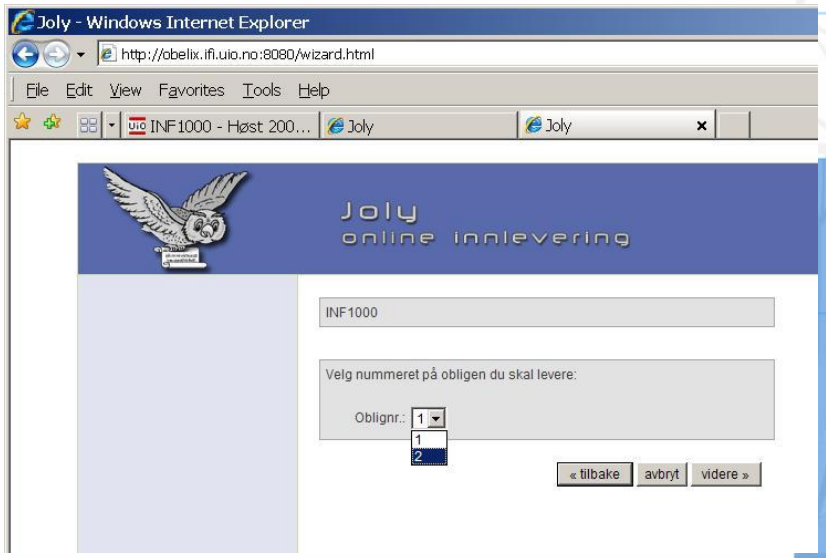
Joly – innlevering av obligatoriske oppgaver

- Link fra kurssiden vil komme opp til:

http://obelix.ifi.uio.no:8080/

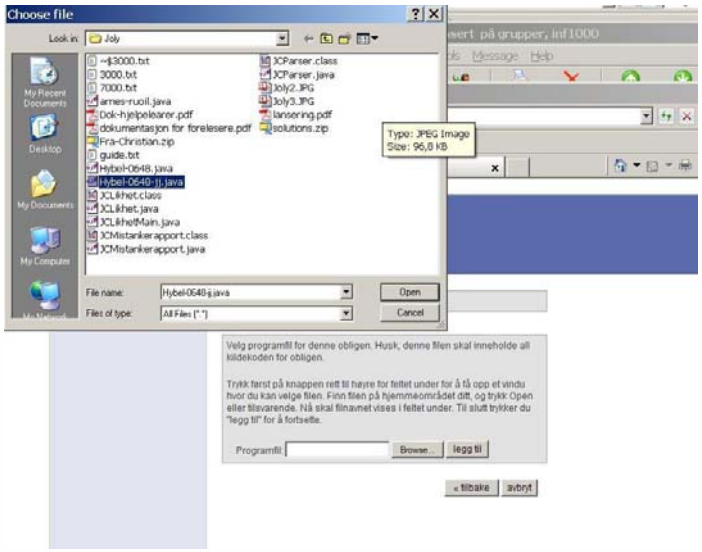
1) Velg kurs (INF1000)

2) Velg oblig

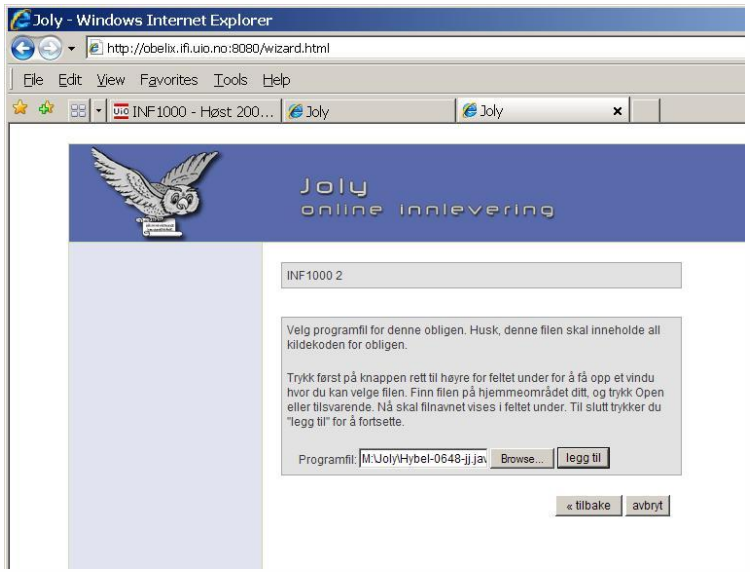


3a) Velg fil til oblig (hele stien med filnavnet sist)

3b) Velg fil til oblig (Trykk 'Browse' og let etter filen)

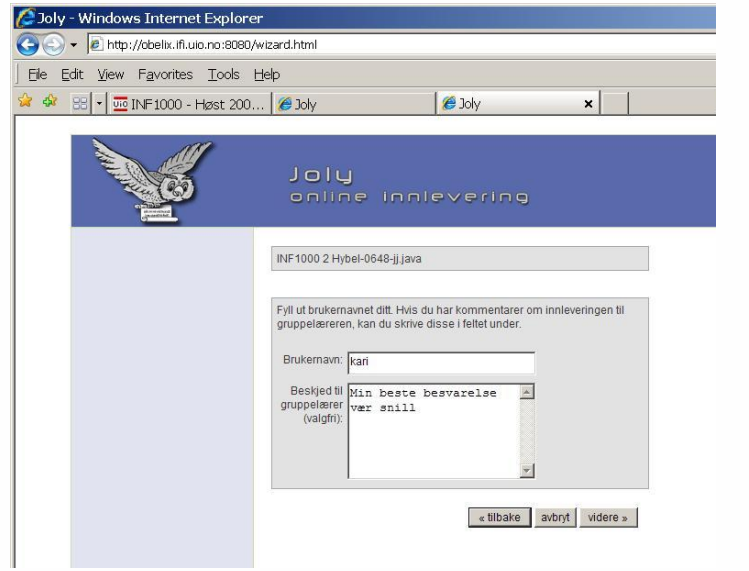


3c) Velg fil til oblig (Trykk ''Open' og så 'legg til')



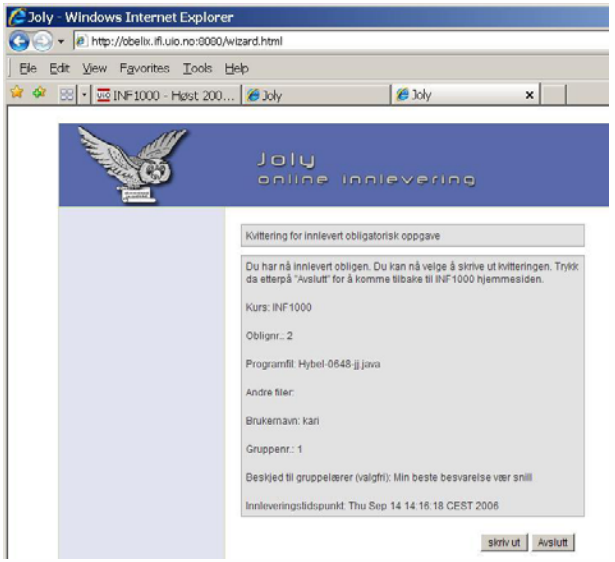
4) Velg evt. tilleggsfil til oblig (ikke vanlig, trykk bare videre)

5) Skriv brukernavn og evt. beskjed til gruppelærer.



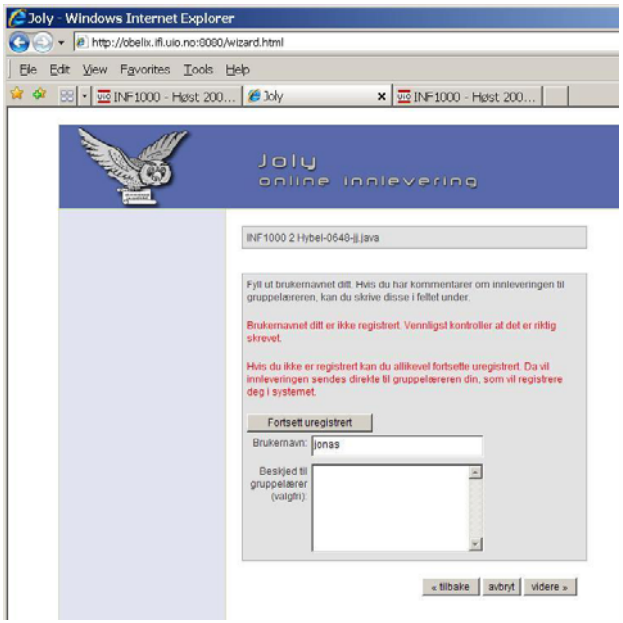
6) Sjekk opplysninger, trykk 'send inn'.

7) Kvittering, skriv ut og avslutt.



Hva hvis ?

- Jeg skal sende inn forbedret oblig
  - OK – alle blir lagret (og gruppelærer retter selvsagt siste)
- Hvordan legger jeg inn gruppenummer
  - Ikke nødvendig. Systemet finner selv hvilken gruppe du er påmeldt (og skal sende obliger til)
- Joly-systemet er nede når jeg skal levere
  - Levér på vanlig e-post til din gruppelærer (så legger han/hun) inn obligen din
- Hvis jeg ikke er registrert som student på gruppa
  - Da kan du fortsatt levere, men besvarelsen blir ikke lagret i basen
  - se neste foil
- Kan jeg levere hjemmefra
  - Sannsynligvis hvis du har VPN



Repetisjon – Program

- **Setninger** som utføres i **sekvens**, ovenfra og nedover
- **Deklarasjoner**
- **Tilordninger**, som inneholder **uttrykk**
- Andre setninger, som utskrift

- Skrives i en fil, kompileres og kjøres.
- Skrevet i et programmeringsspråk

Repetisjon – Program

```
class Repetisjon {
  public static void main(String args[]){
      double PI = 3.14;
      double radius = 2.0;
      double areal;
      String FORTEKST =
            "Arealet til en sirkel med radius ";

      // Utregning
      areal = PI * radius * radius;

      System.out.println(FORTEKST + radius +
            " ER " + areal + ".");
  }
}
```

Prosedyren ”main”

Alt inne i klasser

En kommentar

Setninger avsluttes med semikolon

Repetisjon – Variable og uttrykk

- En variabel er en plass i maskinens lager (minne)

- Variable må ha **navn**
  - Slik at vi kan angi i programmet vårt hvilken plass i lageret vi snakker om

- Variable må ha **type**
  - Så vi vet hvor stor plass variabelen tar og hva det er som ligger der

Utrykk

- Aritmetiske uttrykk

- Logiske uttrykk:

UNIVERSITETET I OSLO

Presedensregler for logiske uttrykk

- Presedens er regler for hvilken rekkefølge utregninger skal skje i et uttrykk
- Vi snakker om høyere og lavere presedens

- Høyest: Metodekall
- !
- <, <=, >, >=
- ==, !=
- &&
- ||

UNIVERSITETET I OSLO

Presedens – utføring

- Setter inn parenteser rundt alle deler av uttrykket
- Hver del har kun én operator, slik: ((*op1*) + (*op2*))
- Vi "pakker inn" hvert uttrykk, men "lukker" rundt de med høyere presedens først og fortsetter med lavere presedens. For hvert nivå tar vi for oss operatorene fra venstre mot høyre.
- De vi har satt parentes rundt kan vi anta at er veldefinert. De kan regnes ut til en verdi etter presedensreglene, for de har fått parenteser.

- Regner så ut ved å lese fra venstre mot høyre og regner ut for hver sluttparentes), altså regnes begge sider i et deluttrykk ut før verdien av deluttrykket regnes ut med operatoren.

UNIVERSITETET I OSLO

Aritmetisk uttrykk – Sette 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))
```


UNIVERSITETET I OSLO

Aritmetisk uttrykk – Beregne uttrykk

```
(((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?
```


Logisk uttrykk – Sette 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++))))
 (u>2) || ((v<2) && (b == (!((++u) == (v++)))))
((u>2) || ((v<2) && (b == (!((++u) == (v++))))))
```


Logisk uttrykk – Beregne uttrykk

```
((u>2)  || ((v<2) && (b == (!((++u) == (v++))))))
( false || ((v<2) && (b == (!((++u) == (v++))))))
( false || (true  && (b == (!((++u) == (v++)))))) u: 1, v: 1
( false || (true  && (b == (!(    2 == (v++)))))) u: 2, v: 1
( false || (true  && (b == (!(    2 == 1 )))))  u: 2, v: 2
( false || (true  && (b == (! false ))))
( false || (true  && (b == true )))
( false || (true  && true ))
( false || true)
  true
```


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 – Sette 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 – Beregne uttrykk

```
((((“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”
```


Presedens – Oppsummert

- Det kan være vanskelig å se hvilken rekkefølge det vil bli utført.
- Bruk parenteser for mye heller enn for lite
- Helst ikke bruk ++ og -- inne i lengre uttrykk
- Se på eksempler og oppgaver og øv på bruk av uttrykk

Hvorfor ikke alltid bruke double?

- Mens regning med heltall alltid er eksakt, er regning med desimaltall ikke det - maskinen kan gjøre avrundingsfeil, slik som her:
```
double x = 0.1;
double y = (x + 1) - 1;
// Nå er verdien til x == y false!
```
- Verdiene til x og y er nesten like, men fordi det er en forskjell i et av desimalene langt ute blir x==y false. Slike avrundingsfeil betyr ofte veldig lite, men du kan ikke stole på at alle desimalene er korrekte når du regner med double.
- Det tar mer plass i hukommelsen å holde en double-verdi enn å holde en int-verdi.
- Det kan ta mer tid å gjøre beregninger med desimaltall enn med heltall.
- Konklusjon: når det er naturlig å bruke heltall bruker vi int og når det er naturlig å bruke desimaltall bruker vi double

UNIVERSITETET I OSLO

Matematiske funksjoner - Avrunding

```
class Avrunding {
  public static void main (String [] args) {
      double x = 0.53;
      // Avrund nedover:
      System.out.println((int) Math.floor(x));
      // Avrund oppover:
      System.out.println((int) Math.ceil(x));
      // Avrund til nærmeste heltall:
      System.out.println((int) Math.round(x));
  }
}
```


UNIVERSITETET I OSLO

Resultat

```
$ javac Avrunding.java
$ java Avrunding
0
1
1
$
```


UNIVERSITETET I OSLO

Kompileringsfeil og kjøretidsfeil

- Kompileringsfeil
  - Feil som oppdages av **javac**
  - Feilformulerte setninger
  - Feil type
  - Programmet blir ikke kompilert
  - Husk: Tidligere kompilerte utgaver kan ligge der
- Kjøretidsfeil
  - Feil som oppdages av **java**
  - Feil vi ikke kunne vite om før programmet ble kompilert
  - Programmet ”krasjer”
- Designfeil
  - Bruk av feil formel eller fremgangsmåte. Resultatet blir feil.

UNIVERSITETET I OSLO

Kompileringsfeil

```
class FeilType {
    public static void main(String args[]){
          int i = 123456;
          boolean b;
          b = i;
    }
}
```

```
$ javac FeilType.java
FeilType.java:5: incompatible types
found   : int
required: boolean
          b = i;
              ^
1 error
```

Kjøretidsfeil

```
class DivNull {
    public static void main(String args[]){
        int x = 7;
        int y = 0;
        int z = x/y;
    }
}
```

```
$ javac DivNull.java
$ java DivNull
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at DivNull.main(DivNull.java:5)
```


Hvordan løse oppgaver

1. **Bestem programmets oppførsel sett utenfra:**
   1. Hva skal være inndata (input) til programmet?
   2. Hvordan skal programmet få tak i inndataene?
   3. Hva skal være utdata (output) fra programmet?
   4. Hvordan skal utdataene presenteres for brukeren?
2. **Avgjør hvordan du skal transformere inndata til utdata:**
   1. Hvordan skal inn- og utdata representeres (lagres) i programmet?
   2. Reduser transformasjonen inndata -> utdata til en sekvens av trinn hvor hvert trinn gjør en enkel ting med dataene og hvor hvert trinn er enkelt å programmere.
3. **Skriv programkode (og test løsningen).**

Eksempel: Celsius og Fahrenheit

- Problem:
  I Norge angis vanligvis temperaturer i Celsius (C), mens man bl.a. i USA benytter Fahrenheit (F). F.eks. svarer 0 C til 32 F.

  Lag et program som lager en tabell som nedenfor (og med temperaturer i Fahrenheit fylt inn):

Celcius	Fahrenheit
-10.0	.....
0.0	.....
37.0	.....
100.0	.....

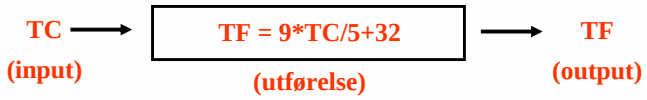
Hvilke data beskriver problemet?

- Inndata:
  - De fire Celcius-temperaturene -10, 0, 37 og 100 (desimaltall)
  - Vi tenker oss at temperaturene er gitt når vi skriver programmet. Senere skal vi se hvordan programmet kunne ha lest inndata fra terminal (fra brukeren).
- Utdata:
  - De tilsvarende (konverterte) Fahrenheit-temperaturene (desimaltall)
  - Skal skrives ut på skjermen i en tabell

Transformere inndata til utdata

- Vi må kjenne formelen for å regne om fra Celcius til Fahrenheit. La
  TC = Temperatur i Celcius
  TF = Temperatur i Fahrenheit
- Vi finner i et oppslagsverk at omregningsformelen er

  TF = 9 * TC / 5 + 32
- Dermed blir fremgangsmåten slik:



Programskisse – "Pseudokode"

```
class TemperaturKonvertering {
   public static void main (String[] args) {
      <deklarasjoner>
      <Skriv overskrift>

      <sett TC lik -10>
      <regn ut TF>
      <skriv ut>

      <sett TC lik 0>
      <regn ut TF>
      <skriv ut>

      <sett TC lik 37>
      <regn ut TF>
      <skriv ut>
      <sett TC lik 100>
      <regn ut TF>
      <skriv ut>
   }
}
```


Ferdig program

```
class TemperaturKonvertering {
   public static void main (String[] args) {
        double TC, TF;
        System.out.println("Celcius Fahrenheit");

        TC = -10;
        TF = 9 * TC / 5 + 32;
        System.out.println(TC + " " + TF);

        TC = 0;
        TF = 9 * TC / 5 + 32;
        System.out.println(TC + " " + TF);

        TC = 37;
        TF = 9 * TC / 5 + 32;
        System.out.println(TC + " " + TF);

        TC = 100;
        TF = 9 * TC / 5 + 32;
        System.out.println(TC + " " + TF);
   }
}
```


Innlesning fra terminal

- Innlesning fra terminal kan gjøres på flere måter i Java. I INF1000 bruker vi pakken easyIO. Du må da skrive i toppen av programmet:

```
import easyIO.*;
```

- Inne i klassen skriver vi følgende før vi kan starte innlesning:

```
In tastatur = new In();
```

- Så kan vi lese inn fra terminal (=tastatur), f.eks. et heltall:

```
int radius;
System.out.print("Oppgi radiusen: ");
radius = tastatur.inInt();
```

Eksempel

Vi importerer pakken easyIO.

Vi oppretter en verktøykasse for lesing fra terminal og lager en variabel tast som blir vårt håndtak til denne verktøykassen.

```
import easyIO.*;

class LesFraTerminal {
  public static void main (String [] args) {
      In tast = new In();
      System.out.print("Skriv et heltall: ");

      int k = tast.inInt();

      System.out.println("Du skrev: " + k);
  }
}
```

I verktøykassen ligger det bl.a. en metode for å lese et heltall fra terminalen.

Resultat

```
$ javac LesFraTerminal.java
$ java LesFraTerminal
Skriv et heltall: 123
Du skrev: 123

$
```


Lesemetoder

```
// 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

- Metodene **inInt()**, **inDouble()** og **inWord()** virker slik:
  - De hopper over eventuelle innledende blanke tegn.
  - De leser så alt fram til neste blanke tegn eller linjeskift. Dersom det som leses ikke er et heltall når inInt() brukes eller et desimaltall når inDouble() brukes, gis det en feilmelding og man får en ny sjanse (3 sjanser).
- Metoden **inChar()** virker slik:
  - Den leser neste tegn, enten det er et blankt tegn eller ikke.
- Metoden **inLine()** virker slik:
  - Den leser alt fram til slutten av linjen (inkludert blanke tegn), men ignorerer linjer hvor det kun står (igjen) et linjeskift.

Hvordan lesemetodene virker

Terminal-input: `_ _ x y z _ 1 6 1 2 7 5` `_ = blank`

`String s1 = tast.inWord();` `String s2 = tast.inWord();`	`s1: "xyz"` `s2: "161275"`
`String s1 = tast.inWord();` `int x = tast.inInt();`	`s1: "xyz"` `x : 161275`
`String s = tast.inLine();`	`s: "  xyz 161275"`

`char c1 = tast.inChar();`
`char c2 = tast.inChar();`
`char c3 = tast.inChar();`
	`c1: ' '`
	`c2: ' '`
	`c3: 'x'`

`int x = tast.inInt();`	`feilmelding`

Eksempel: lese data om en person

- Problem:
  Lag et program som leser fra terminal disse dataene om en person:
  - Navn
  - Yrke
  - Alder

  og som skriver ut dataene på skjermen etterpå.
- Framgangsmåte:
  - Vi bruker **inLine()** til å lese navn og yrke, og **inInt()** til å lese alder.

Ferdig program

```
import easyIO.*;
class LesDataOmPerson {
   public static void main (String [] args) {
        String navn, yrke;
        int alder;

        In tast = new In();
        System.out.print("Navn: ");
        navn = tast.inLine();

        System.out.print("Yrke: ");
        yrke = tast.inLine();

        System.out.print("Alder: ");
        alder = tast.inInt();

        System.out.print("Hei " + navn + ", du er " +
                alder);
        System.out.println(" år gammel og jobber som " +
                yrke);
   }
}
```

Et eksempel til

```
import easyIO.*;

class LesInnOrd {
   public static void main (String [] args) {
       In tastatur = new In();
       System.out.print("Skriv tre ord: ");
       String s1 = tastatur.inWord();
       String s2 = tastatur.inWord();
       String s3 = tastatur.inWord();

       System.out.println(
             "Du skrev disse ordene:");
       System.out.println("  " + s1);
       System.out.println("  " + s2);
       System.out.println("  " + s3);
   }
}
```


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. For å få tilgang til denne pakken må vi som før skrive helt i toppen av programmet vårt (**før** class):

```
import easyIO.*;
```

- Inne i klassen skriver vi følgende før vi kan starte formatert utskrift:

```
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
```

UNIVERSITETET I 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
```

- 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
```