INF1000- uke 3

Litt om Java-teknologien
Fortsette innføringen i språket Java
Idag: variable, uttrykk, lese fra terminal, forgreninger

6. september 2005

Arne Maus
Universitetet i Oslo

Java

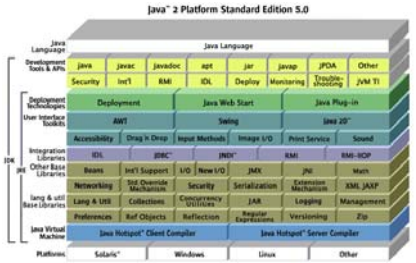
- **Java** er navnet på et programmeringsspråk – og navnet på hele det programsystemet som må være installert på maskinen for å kompilert og kjørt Java-programmer.
- Java ble lansert i 1995 av dataselskapet Sun Microsystems, og er i dag blant de mest populære programmeringsspråk (men slett ikke det eneste!).
- Java er gratis og tilgjengelig fra http://java.sun.com (og fra Ifi-CD'en).
- Versjonen som brukes nå kalles *Java 5 plattformen*.

Ulike varianter for ulike behov

Standard Edition (J2SE)

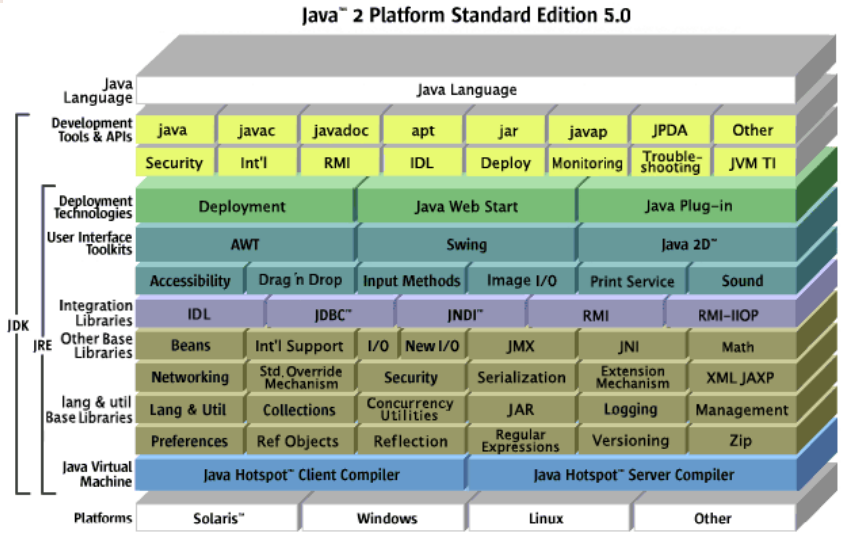
To sentrale begreper:

- **Java Runtime Environment (JRE)**
  System for å kjøre kompilerte Java-programmer.
- **Software Development Kit (SDK)**
  JRE + programmer for å kompilere, feilsøke ("debugge") og dokumentere Java-programmer.



I tillegg til dette kan man legge til egne "pakker" som gir økt funksjonalitet.
I INF1000 brukes det en slik pakke, med navn easyIO.

Standard Edition (J2SE)



Installasjon av Java på egen maskin

- Installer J2SE SDK
  - Fås fra Ifi-CD'en (fås på øvingsgruppene)
  - ...eller fra http://java.sun.com/j2se/1.5.0/download.jsp
  - Klikk på JDK 5.0 Update 4 (eller enda nyere versjon):
    - **Download JDK 5.0 Update 4**
    - Klikk av at du aksepter lisensbetingelsene og velg riktig operativsystem
  - Kjører du Windows, kan du også se under lærebokas hjemmeside: http://www.universitetsforlaget.no/java/java.php
- Installer easyIO
  - Hentes fra www.universitetsforlaget.no/java
  - (alternativt fra kursets hjemmeside: easyIOkode og kilde)

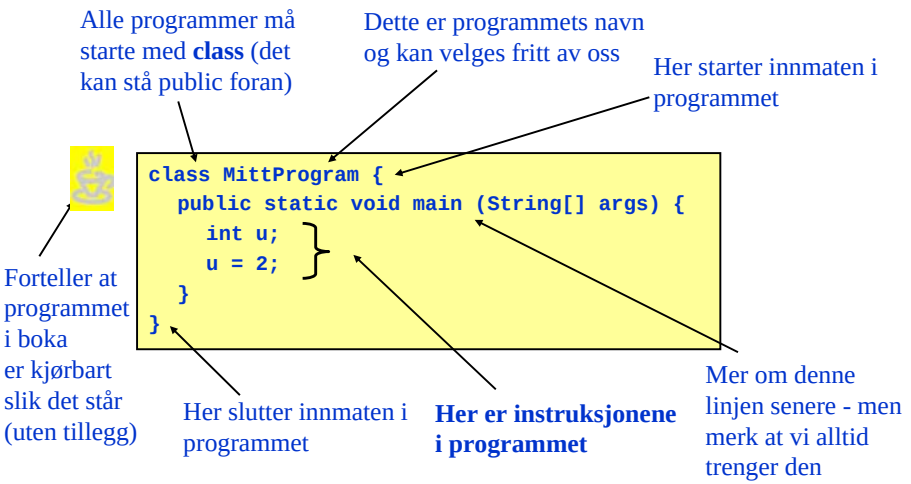
Kompilere og kjøre Java-programmer

For flere detaljer: http://java.sun.com/docs/books/tutorial/getStarted/cupojava/

- **Unix:**
  - Start et terminalvindu (xterm-vindu)
  - Endre filområde (directory) til der programfilen ligger
  - For å kompilere: javac MittProgram.java
  - For å kjøre: java MittProgram
- **Windows:**
  - Start et kommandovindu ved å gå inn i Start-menyen og velge
    - `MS-DOS Prompt` (Window 95 og 98)
    - `Command Prompt` (Windows NT, 2000 og XP)

    (hvis skjult, se under All Programs / Accessories eller tilsvarende)
  - Resten som for Unix-baserte plattformer.
- **Mac OS:**
  - For å kompilere: legg programfil-ikonet oppå javac-ikonet (se i mappen MRJ SDK 2.2/Tools/JDK Tools eller liknende).
  - For å kjøre: legg ikonet for den kompilerte filen (.class-filen) oppå ikonet JBindery (se i mappen MRJ SDK 2.2/Tools/Application Builders/JBindery).

Bestanddelene i et Java-program



Hva er vitsen med class?

- En setning av typen

```
class {
  <...masse rart...>
}
```

kalles en klassedeklarasjon (eller bare klasse).

- Tenk på en klasse som en samling data (tall, tekst, bilder, osv) og operasjoner som vi ønsker å kunne utføre på dataene.
- Senere i kurset kommer hvert program til å bestå av mange klasser. Hver klasse har sitt ansvarsområde: å utføre visse oppgaver, håndtere visse typer data, eller begge deler. Dette gjør bl.a. programmene oversiktlige og gjør det lettere å bruke biter av programmet på nytt i andre sammenhenger.

Repetisjon: datatyper

Datatype	Beskrivelse	Eksempel
int	heltall	`int k = 3;`
double	desimaltall	`double x = 3.14;`
boolean	sannhetsverdi	`boolean b = true;`
char	tegn	`char c = '@';`
String	tekst	`String s = "Hei på deg";`

+ noen flere (short, long, byte, float)

Variabeldeklarasjoner

- Variable kan deklareres hvor som helst i et program, og de kan endres hvor som helst etter at de er deklarert.
- Variable har ingen verdi rett etter en deklarasjon:

```
int lengde;
lengde = lengde + 1;    // Ulovlig!
```

- Vi kan gi variable en verdi når vi deklarerer dem:

```
int lengde = 4;
lengde = lengde + 1;          // Lovlig
```

- Vi kan også vente med å gi en variabel verdi:

```
int lengde;
.....
lengde = 4;
lengde = lengde + 1;          // Lovlig
```


Hvis du glemmer å initialisere en variabel

Forsøk på å kompilere et program med en slik feil:

```
C:\Eksempel.java:4: variable lengde might not have
  been initialized
    lengde = lengde + 1;
             ^
1 error

Tool completed with exit code 1
```

Dette er en veldig vanlig feil, så lær deg å kjenne igjen denne feilmeldingen!

Avsluttende om variable

- Unngå i størst mulig utstrekning å samle mange variabeldeklarasjoner på en linje:

```
int år, måned, dag, alder;
```

Uoversiktlig

```
int år;    // Fødselsår
int måned; // Fødselsmåned (1-12)
int dag;   // Fødselsdato (1-31)
int alder; // Alder i antall år
```

Oversiktlig

- Deklarer variable først når du trenger dem – ingen grunn til å samle alle variabeldeklarasjoner ett sted med mindre de naturlig hører sammen.

Eksempel 1: kompileringsfeil

```
class IkkeRiktig {
   public static void main (String [] args) {
      double x;
      int y;
      x = 2;
      y = x;  // Her prøver vi å sette y lik et desimaltall
   }
}
```

```
FEIL UNDER KOMPILERING

IkkeRiktig.java:6: possible loss of precision
found   : double
required: int
       y = x;
           ^
1 error
```


Eksempel 2: kompileringsfeil

```
class MerFeil {
   public static void main (String [] args) {
      boolean b;
      b = 2;   // Her prøver vi å sette b lik et heltall
   }
}
```

```
FEIL UNDER KOMPILERING

MerFeil.java:4: incompatible types
found   : int
required: boolean
       b = 2;
           ^
1 error
```


Eksempel 3: kompileringsfeil

```
class EndaMerFeil {
   public static void main (String [] args) {
      int x, y;
      y = x;   // Her prøver vi å sette y lik en udefinert verdi
   }
}
```

```
FEIL UNDER KOMPILERING

EndaMerFeil.java:4: variable x might not have been initialized
       y = x;
           ^
1 error
```

Eksempel 4: kjørefeil

```
class Kjorefeil {
   public static void main (String [] args)
   {
      int x = 3, y = 0, z;
      z = x / y;  // Her prøver vi å dele på null
   }
}
```

Test programmet

```
FEIL UNDER KJØRING

Exception in thread "main" java.lang.ArithmeticException: / by zero
                at Kjorefeil.main(Kjorefeil.java:4)
```


Konvertering

- Når det er nødvendig vil Java automatisk (implisitt) konvertere heltall til desimaltall, som f.eks. i disse tre tilfellene:

```
double x = 7;

int a = 15;
   double x = a;

c) double x = (7 + 14) * 3 - 12;
```

- Derimot vil Java ikke automatisk konvertere desimaltall til heltall, siden det generelt fører til en endring i verdien:

```
int a = 7.15;                  // Ikke lov!!

double x = 15.6;
   int a = x;                  // Ikke lov!!

int a = 3.14 * 7 / 5;          // Ikke lov!!
```


Konvertering *forts.*

- Dersom vi virkelig ønsker å konvertere et desimaltall til et heltall, må vi eksplisitt be om det:

```
int a = (int) 7.15;              // Lovlig!

double x = 15.6;
   int a = (int) x;              // Lovlig!

   int a = (int) 3.14 * 7 / 5;   // Lovlig!
```

- I noen tilfeller - når tallene allikevel er hele - spiller det ingen rolle om man bruker int eller double. Så hvorfor ikke alltid bruke double?

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 du int og når det er naturlig å bruke desimaltall bruker du double!

Avrunding

- Konvertering fra desimaltall til heltall involverer normalt en 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));
  }
}
```


Heltallsdivisjon

- Java konverterer ikke fra heltall til desimaltall når to heltall adderes, subtraheres, multipliseres eller divideres:
  - 234 + 63 : heltall (int)
  - 235 - 23 : heltall (int)
  - 631 * 367 : heltall (int)
  - 7 / 2 : heltall (int)
- Legg spesielt merke til siste punkt ovenfor:

Når to heltall divideres på hverandre i Java blir resultatet et heltall, selv om vanlige divisjonsregler tilsier noe annet. Dette kalles heltallsdivisjon, og resultatet er det samme som om vi fulgte vanlige divisjonsregler og så avrundet nedover til nærmeste heltall. Dvs (7/2) = = (int) (7.0/2.0) = = 3.

Konkatenering av tekst

Det er ofte nyttig å slå sammen (=konkatenere) flere tekstbiter til en stor tekstbit før vi skriver ut på skjerm. Det kan vi gjøre i Java med + slik som i dette eksemplet:

```
class SkrivPaaSkjerm {
  public static void main (String [] args) {
    double hastighet = 90.5;
    double avstand = 360.2;
    System.out.println("Kjørelengde: " + avstand + " km");
    System.out.println("Hastighet: " + hastighet + " km/t");
    System.out.println("Kjøretid: " + avstand/hastighet + " timer");
  }
}
```

NB: husk at + også brukes til å addere tall. Det er forskjell på

System.out.println("2" + "3"); (utskriften blir: 23)
System.out.println("2 + 3"); (utskriften blir: 2 + 3)
System.out.println(2 + 3); (utskriften blir: 5)

Oppgave

- Avgjør i hvert tilfelle hvilken datatype resultatet har:

Uttrykk	Datatype
2 + 6 * 3	
14.2 + 6	
3/2 + 4	
"Vekt: " + 25 + " kg"	
"" + 17.4	
Math.ceil(5.3) + (int) 3.25	

java.lang
Class Math

static double ceil(double a)
Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.

Når du løser oppgaver

1. **Bestem programmets oppførsel sett utenfra:**
   - Hva skal være inndata (input) til programmet?
   - Hvordan skal programmet få tak i inndataene?
   - Hva skal være utdata (output) fra programmet?
   - Hvordan skal utdataene presenteres for brukeren?
2. **Avgjør hvordan du skal transformere inndata til utdata:**
   - Hvordan skal inn- og utdata representeres (lagres) i programmet?
   - 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: Celcius og Fahrenheit

- Problem:

  I Norge angis vanligvis temperaturer i Celcius (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

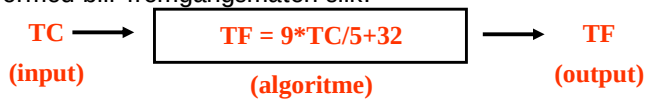
- Vi må kjenne formelen for å regne om fra Celcius til Fahrenheit. La

  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

```
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.*;
```

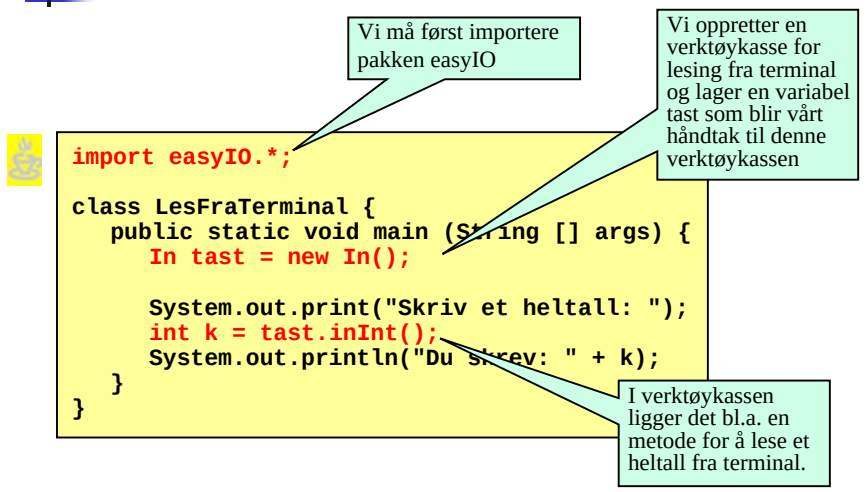
- Inne i klassen skriver vi følgende før vi kan starte innlesning:

```
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



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();
```


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.
- 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) Hvis innlesningen står på slutten av linja, leses hele neste ikke-tomme linje (en tom linje er en som bare består av linjeskift – ingen data).

Hvordan lesemetodene virker, 5 forsøk på å lese samme data fra tastaturet

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`

Hvilken lesemetode skal jeg velge?

- Først: importere easyIO og åpne forbindelse til tastaturet
- Lese item for item:
  - For å lese et heltall: inInt()
  - For å lese et desimaltall: inDouble()
  - For å lese ett ord: inWord()
  - For å lese alle ord: inWord("\n")
- Lese linje for linje:
  - For å lese resten av linja: inLine()
  - (For også få med seg tomme linjer : readLine();)
- Lese tegn for tegn:
  - For å lese neste tegn (også hvite tegn): inChar()

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 LesFraTerminal2 {
   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);
   }
}
```


Programmer med forgreninger

- En svært nyttig programmeringsteknikk er å bruke forgreninger, dvs forskjellige instruksjoner utføres i ulike situasjoner.
- Vi kan få til dette med en if-setning:

```
if (logisk uttrykk)
{
  <instruksjoner>
}
else
{
  <instruksjoner>
}
```

f.eks. x < y eller et annet uttrykk som enten er true eller false

denne blir utført når det logiske uttrykket er sant (true)

denne blir utført når det logiske uttrykket er usant (false)

- Eksempel:

```
if (x > 0) {
  System.out.println("Tallet er positivt");
} else {
  System.out.println("Tallet er ikke positivt");
}
```

Programmer med forgreninger

- Else-delen kan utelates, slik som her:

```
if (pris > 1500) {
  System.out.println(”Det er for dyrt”);
}
```

- Vi kan legge if-setninger inni if-setninger:

```
if (lønn < 400000) {
  if (ferieuker < 8) {
    System.out.println(”Ikke søk på jobben”);
  }
}
```

- Vi kan lage sammensatte if-setninger av typen

```
if (a < 10) { // a er positivt heltall
  System.out.println(”Ett siffer”);
} else if (a < 100) {
  System.out.println(”To siffer”);
} else {
  System.out.println(”Mer enn to siffer”);
}
```


Eksempel på bruk av if-setning

Program som avgjør hvem av to personer som er høyest:

```
import easyIO.*;

class Hoyde {
  public static void main (String[] args) {
    In tastatur = new In();
    double høyde1, høyde2;

    System.out.print("Høyden til Per: ");
    høyde1 = tastatur.inDouble();
    System.out.print("Høyden til Kari: ");
    høyde2 = tastatur.inDouble();

    if (høyde1 > høyde2) {
      System.out.println("Per er høyere enn Kari");
    } else {
      System.out.println("Per er ikke høyere enn Kari");
    }
  }
}
```


Eksempel: Body Mass Index

Oppgave:

Body Mass Index (BMI) er et mål som kan regnes ut fra høyden og vekten til en person. Ifølge verdens helseorganisasjon (WHO)[1] :

BMI	Vektstatus
Under 18.5	Undervekt
18.5 – 24.9	Normal vekt
25.0 – 29.9	Overvekt
30.0 eller høyere	Fedme

Vi skal lage et program som beregner BMI ut fra høyde og vekt og gir melding om hvilken vektstatus (se tabellen) det tilsvarer.

[1]Se http://www.who.int/hpr/NPH/docs/gs_obesity.pdf

Inndata og utdata

- **Inndata:**
  - Personens høyde (i m)
  - Personens vekt (i kg)
  - Leses fra terminal
- **Utdata:**
  - BMI
  - Skrives ut på skjerm, sammen med en av beskjedene
    - *Undervekt* (hvis BMI <= 18.4)
    - *Normal vekt* (hvis 18.5 <= BMI <= 24.9)
    - *Overvekt* (hvis 25.0 <= BMI <= 29.9)
    - *Fedme* (hvis BMI >= 30.0)

Transformere inndata til utdata

- Vi må kjenne formelen for å regne ut BMI. La

  vekt = personens vekt i kg
  hoyde = personens høyde i m

  Da er

  BMI = vekt / (hoyde*hoyde)

Første versjon – hvilke handlinger

```
import easyIO.*;

class BMI1{
    public static void main (String [] args) {
        String navn;
        String vektklasse;
        double vekt,høyde;
        double bmi;
        In tast = new In();

          ... handlinger her.............
    }
}
```



```
System.out.print("Navn: ");
   navn = tast.inLine();
   System.out.print("Høyde:i meter ");
   høyde = tast.inDouble();
   System.out.print("Vekt: ");
   vekt = tast.inDouble();

   bmi = vekt/(høyde*høyde);

   if (bmi < 18.4)
     vektklasse = "undervektig";
   else if (bmi <= 24.9)
     vektklasse = "normal";
   else if (bmi <=29.9)
     vektklasse = "overvektig";
   else
      vektklasse = "fet";

   System.out.println(navn +" med vekt:" +vekt +" og høyde:"
   + høyde + " har BMI = " +
   bmi +", og er " + vektklasse);
```

BMI2.java

```
System.out.print("Navn: ");
   navn = tast.inLine();
   System.out.print("Høyde:i meter ");
   høyde = tast.inDouble();
   System.out.print("Vekt: ");
   vekt = tast.inDouble();

   bmi = vekt/(høyde*høyde);

   if (bmi < 18.4)
     vektklasse = "undervektig";
   else if (bmi <= 24.9)
     vektklasse = "normal";
   else if (bmi <=29.9)
     vektklasse = "overvektig";
   else
      vektklasse = "fet";

   System.out.println(navn +" med vekt:" +vekt +
     " og høyde:" +høyde + " har BMI = "
      + bmi +", og er " + vektklasse);


    System.out.println(navn +" med vekt:" +vekt +
      " og høyde:" + høyde + " har BMI = "
      + Format.format(bmi,2)  +", og er " + vektklasse);
```

Slik kan for mange sifre etter komma fjernes med Format.format(doble, int)

Alternativ til if-else: switch

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

Et uttrykk som gir en verdi som er av en av typene char eller int (evt. byte eller short)

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

- Nøkkelordet break avbryter utførelsen av switch-setningen. Når break mangler, fortsetter utførelsen på neste linje (det er sjelden ønskelig).

Eksempel

```
class BrukAvSwitch {
  public static void main (String [] args) {
    char c = 'x';

    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");
    }
  }
}
```