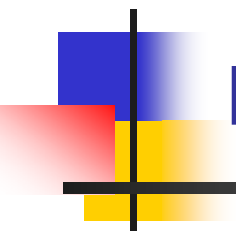


INF1000 (Uke 3)

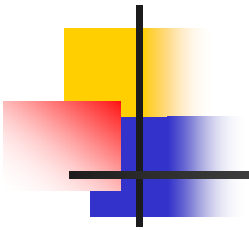
Mer om uttrykk, terminal I/O, forgreninger

Grunnkurs i programmering

Institutt for Informatikk

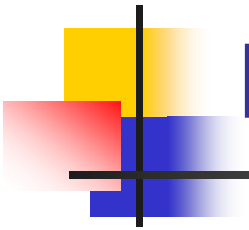
Universitet i Oslo

Are Magnus Bruaset og Anja B. Kristoffersen



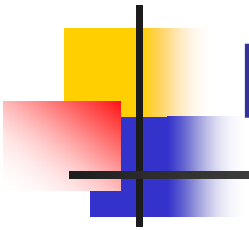
I dag

- Litt repetisjon
- Mer om uttrykk
- Lesing og skriving til terminal
- Forgreninger



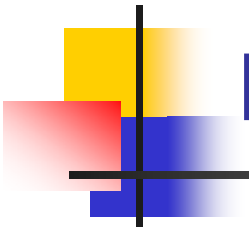
Rep: Datatyper

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



Rep: Oppsummering - variabler

- En variabel deklareres i programmet, og vi får da en plass i lageret med
  - Et navn
  - En type
  - En verdi

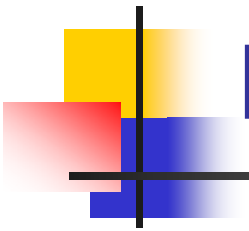


Rep: Oppsummering - variabelnavn

- Navn på variabler bør begynne med liten bokstav
- Stor bokstav for hver stavelse (**`antallBarn`**)
- Kan ikke være det samme som noen av de reserverte ordene i Java-språket

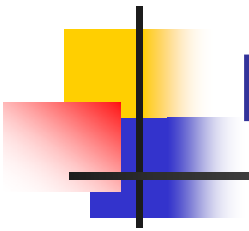
Rep: Oppsummering - tilordning

- Eksempel: **`i = j*3+i ;`**
- Venstre side (**`i`**) er navnet på en variabel som skal få ny verdi
- Høyresiden **`(j*3+i)`** er et regnestykke
  - Variabelnavn (som **`j`** og **`i`**) erstattes med deres nåværende (gamle) verdi



Rep: Oppsummering - uttrykk

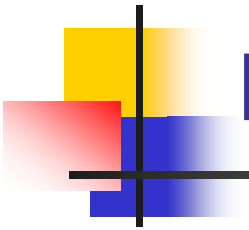
- Tre typer uttrykk som ofte forekommer:
  - Numeriske uttrykk
  - Logiske uttrykk
  - Streng-uttrykk



Rep: Pakker og Java-klassebibliotek

```
double d = Math.sqrt(23);
```

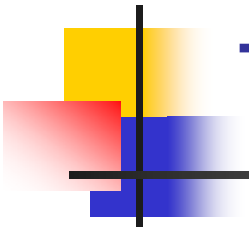
- I et eget Java-bibliotek er det over 2000 ferdigprogrammerte klasser
- Med unntak av en samling av mye brukte klasser, (pakken ”java.lang”), må vi importere en slik samling av klasser hvis vi vil bruke noen av dem



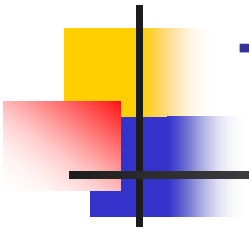
Rep: Pakker og Java-klassebibliotek

- Vi har hittil brukt bibliotekskklassene : **`Math`** og **`System`** som begge er i **`"java.lang"`**
- Skal vi bruke noen andre klasser i andre pakker må disse importeres, f.eks. for IO.
- I toppen av programmet må vi da ha import-setning:

```
import java.io.*;
```



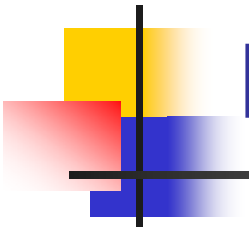
Tilbake på sporet...



Ting å passe på

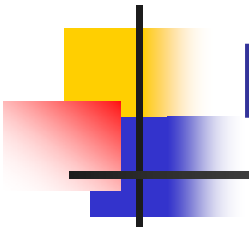
- Multiplikasjon må alltid angis eksplisitt med *:
  - `int prod = 10 a;` `// feil!!`
  - `int prod = 10 * a;` `// riktig`

- Det er forskjell på = og == :
  - = brukes for å sette verdien til en variabel
  - == brukes for å sammenlikne to verdier

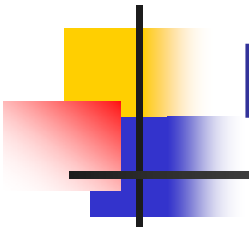


Mer å passe på

- Hvis vi har variabelen `boolean b` så er det ingen forskjell på
  - `b == true`
  - `b`
- Ekstra parenteser kan øke leseligheten for mennesker:
  - `b = x == y;` betyr det samme som `b = (x == y);`



Konvertering

- Hvis nødvendig vil Java automatisk (implisitt) konvertere heltall til desimaltall
- Eksempler:
  - `double x = 7;`
  - ```
    int a = 15;
    double x = a;
    ```
  - `double x = (7 + 14) * 3 - 12;`
    ```



Mer om konvertering

- Java vil **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!!`
    ```

Mer om konvertering

- Kan konvertere et desimaltall til et heltall ved å eksplisitt be om det (casting):

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

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

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

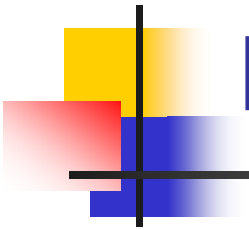
- I noen tilfeller - når tallene allikevel er hele - spiller det ingen rolle om man bruker int eller double. Så hvorfor ikke alltid bruke double?

Hvorfor ikke alltid bruke **`double`**?

- Regning med heltall er eksakt, regning med desimaltall er ikke det - maskinen kan gjøre avrundingsfeil:

```
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`**.



Mer om **`double/int`**

- 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`**!

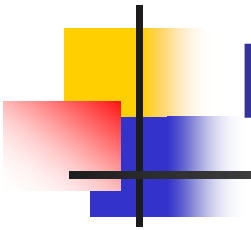
Avrunding

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(Math.round(x));
  }
}
```

Programvareutvikling

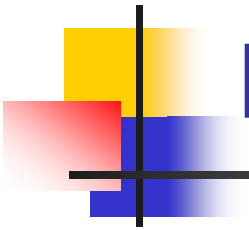
1. Først har vi et problem vi skal løse (en oppgave)
2. Finn hvilke data som beskriver problemet
   - Hva skal være input til programmet (hva er kjent)?
   - Hva skal være output fra programmet (hva ønskes funnet)?
3. Finn en fremgangsmåte (=algoritme) for å løse problemet, dvs. for å komme fra input'ene til output'ene:



Programvareutvikling forts.

(En algoritme er en beskrivelse - ment å leses av mennesker - av hvordan problemet kan løses i flere steg, og hvor hvert steg er lett å oversette til et programmeringsspråk som Java)

4. Skriv et Java-program som "implementerer" løsningen ovenfor, dvs som finner output'ene og f.eks. skriver dem ut på skjerm
5. Til slutt: utfør og test ut programmet

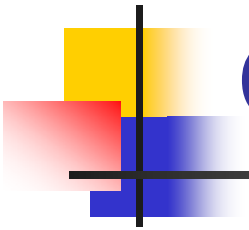


Eksempel: Oppussing

- Problem:

  En vegg er 2.35 x 7.50 meter. Veggen skal males med en maling som dekker 6 kvadratmeter per liter.

  Lag et program som regner ut hvor mange liter maling vi trenger, og hvor mange liters-bokser maling vi må kjøpe.



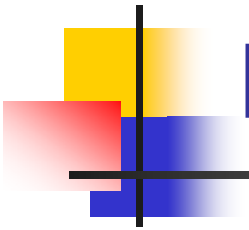
Oppussing forts.

- Formler (algoritme):

  Areal = 2.35 * 7.50

  Antall liter maling = areal / 6

  Antall liters-bokser = antall liter maling, avrundet oppover til nærmeste heltall



Hvilke data beskriver problemet?

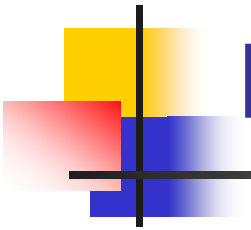
- Input:
  - Veggens dimensjoner (to desimaltall)
  - Hvor stort areal en liter maling holder til (desimaltall)
- Output:
  - Antall liter maling som går med (desimaltall)
  - Antall liters-bokser som trengs (heltall)

Løsningsskisse

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

    <beregn areal>
    <beregn antall liter maling>
    <beregn antall liters-bokser>

    <skriv ut>
  }
}
```

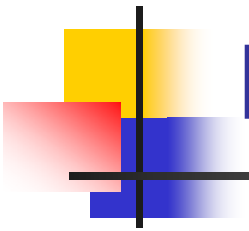
Ting vi må ta stilling til:

- Hvilke variable trenger vi? Hva slags type skal de ha?
- Når skal vi initialisere variablene?
- Hvordan avrunder vi oppover til nærmeste heltall i Java?
- Hvordan skal utskriften se ut?



Løsning

```
class Oppussing {
  public static void main (String[] args) {
    double areal, liter;
    int antBokser;

    areal = 2.35 * 7.50;
    liter = areal / 6;
    antBokser = (int) Math.ceil(liter);

    System.out.println(”Areal: ” + areal);
    System.out.println("Antall liter maling: " + liter);
    System.out.println("Antall liters-bokser: " + antBokser);
  }
}
```



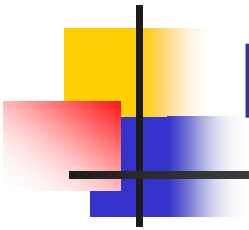
Eksempel: Celcius og Fahrenheit

- 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):

Celsius	Fahrenheit
-10.0	...
0.0	...
37.0	...
100.0	...



Hvilke data beskriver problemet?

- Input:
  - De fire Celsius-temperaturene -10, 0, 37 og 100 (desimaltall)
- Output:
  - De tilsvarende (konverterte) Fahrenheit-temperaturene (desimaltall)

Framgangsmåte

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

  TC = Temperatur i Celsius

  TF = Temperatur i Fahrenheit

Vi finner i et oppslagsverk at omregningsformelen er

TF = 9 * TC / 5 + 32

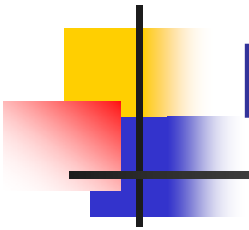
Dermed blir fremgangsmåten slik:

Skisse

```
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>
  }
}
```

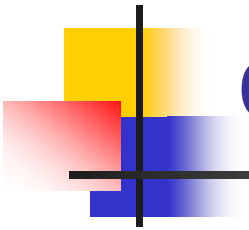
Java-program: endelig løsning

```
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);

  }
}
```



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



easyIO

- Bedre: bruke en ferdiglaget pakke for slikt. I INF1000 bruker vi pakken easyIO. For å få tilgang til denne pakken må vi 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;
// Skriv ut pi: 2 desimaler, høyrejustert på 6 plasser.
skjerm.out(pi, 2, 6);
```

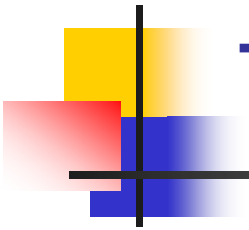
Eksempel

Vi må først importere pakken easyIO

```
import easyIO.*;

class FormatertUtskrift {
   public static void main (String [] args) {
      Out skjerm = new Out();
      int BREDDE1 = 15;
      int BREDDE2 = 30;
      skjerm.out("NAVN", BREDDE1);
      skjerm.outln("ADRESSE", BREDDE2);
      skjerm.out("Kristin Olsen", BREDDE1);
      skjerm.outln("Vassfaret 14, 0773 Oslo", BREDDE2);
   }
}
```

Vi oppretter en verktøykasse for skriving til terminal

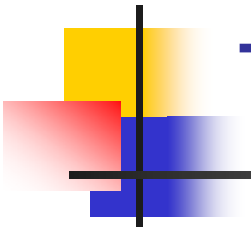
I verktøykassen ligger det bl.a. verktøy (på java-språk: *metoder*) for å skrive til skjerm med og uten linjeskift til slutt.



Tre måter å skrive ut på

- Uten formatering:

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



Tre måter å skrive ut på

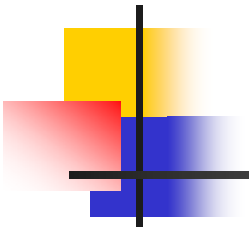
- Angi utskriftsbredde:

```
skjerm.out("Per Hansen", 15);
skjerm.out(12345, 15);
skjerm.out(3.1415, 2, 15);
```

Tre måter å skrive ut på

- Angi utskriftsbredde og justering:

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



Innlesning fra terminal

- Innlesning fra terminal kan gjøres på flere måter i Java. I INF1000 bruker vi pakken easyIO. Som før må vi 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();
```

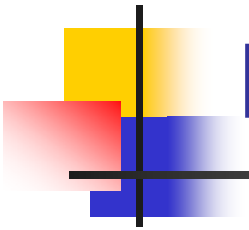
Eksempel

Vi må først importere 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);
   }
}
```

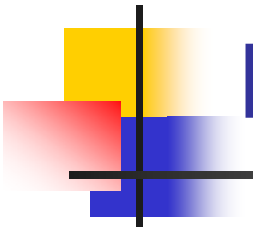
I verktøykassen ligger det bl.a. en metode for å lese et heltall fra terminal.



Lesemetoder

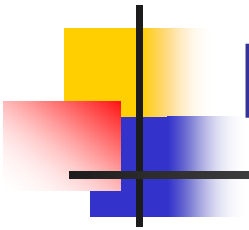
```
// Opprette forbindelse med tastatur:
In tastatur = new In();

// 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()`**:
  - Hopper over eventuelle innledende blanke tegn
  - Leser 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



Hvordan lesemetodene virker

- Metoden **`inChar()`**:
  - Leser neste tegn, enten det er et blankt tegn eller ikke
- Metoden **`inLine()`**:
  - Leser alt fram til slutten av linjen (inkludert blanke tegn)

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();
```

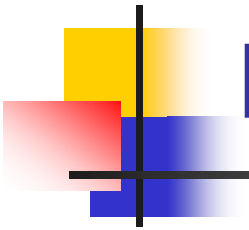
→

```
c1: ' '
c2: ' '
c3: 'x'
```

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

→

```
feilmelding
```



Eksempel: lese persondata

- Problem:

  Lag et program som leser data om en person fra terminal

  - Navn
  - Yrke
  - Alder

  og som skriver ut dataene på skjermen etterpå

- Framgangsmåte:
  - Bruk 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);
  }
}
```

Eksempel 2

```
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);
   }
}
```

Eksempel 3

```
import easyIO.*;

class PigLatin {
   public static void main (String [] args) {
     In tastatur = new In();
     System.out.print("Skriv et ord: ");
     String s = tastatur.inWord();
     String s2 = s.substring(1) + s.charAt(0) + "ay";
     System.out.print("Oversettelse til pig Latin: ");
     System.out.println(s2);
   }
}
```

Programmer med forgreninger

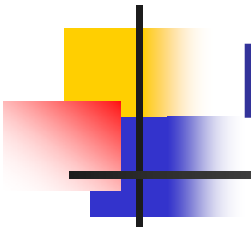
- En svært nyttig programmeringsteknikk er å bruke forgreninger, dvs forskjellige instruksjoner utføres i ulike situasjoner
- Vi kan få til dette med en if-setning:

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

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

Blir utført når det logiske uttrykket er sant (true)

Blir utført når det logiske uttrykket er usant (false)

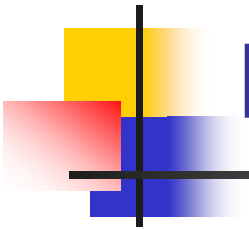


Eksempel

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

Eksempel: Hvem 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: Beregning av skatten

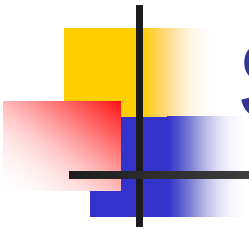
- Problem:

  I det fiktive landet Ruritania er skattereglene slik (beløp i ruritanske kroner):

  10% skatt hvis inntekt < 10 000

  20% skatt hvis inntekt ≥ 10 000

  Lag et program som regner ut, for en gitt inntekt, hvor mye man skal betale i skatt



Skatteregler

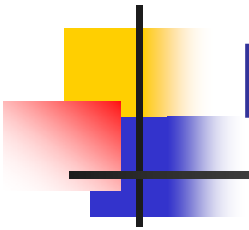
- Formler:

  Skatten man skal betale er

  $0.1 * \text{inntekt}$ hvis $\text{inntekt} < 10\,000$

  $0.2 * \text{inntekt}$ hvis $\text{inntekt} \geq 10\,000$

- Vi er nå klare til å programmere...

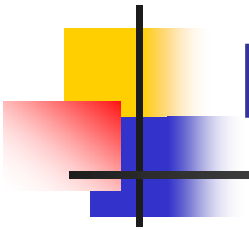


Løsningsskisse

```
import easyIO.*;

class Skatt {
  public static void main (String[] args)
{
    <deklarasjoner>

    <les inntekten fra terminal>
    <beregn skatten>
    <skriv ut resultatet>
  }
}
```

Faktorer vi må ta stilling til

- Hvilke variable trenger vi?
  Hva slags type skal de ha?
- Hvordan beregner vi skatten?
- Hvordan skal utskriften se ut?



Løsning

```
import easyIO.*;

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









  }
}
```

Løsning

```
import easyIO.*;

class Skatt {
  public static void main (String [] args) {
    double inntekt, skatt;
    In tastatur = new In();
    System.out.print("Inntekt: ");
    inntekt = tastatur.inDouble();

    if (inntekt < 10000)
      skatt = 0.1 * inntekt;
    else
      skatt = 0.2 * inntekt;

    System.out.println("Skatt: " + skatt);
  }
}
```