UNIVERSITETET I OSLO

Velkommen til INF1000 høst 2010

Kursansvarlige:

Ragnhild Kobro Runde

Arild Waaler

Forelesning 1:

- Velkommen til kurset!
- Litt praktisk informasjon
- Noen enkle programmer

Første forelesning!

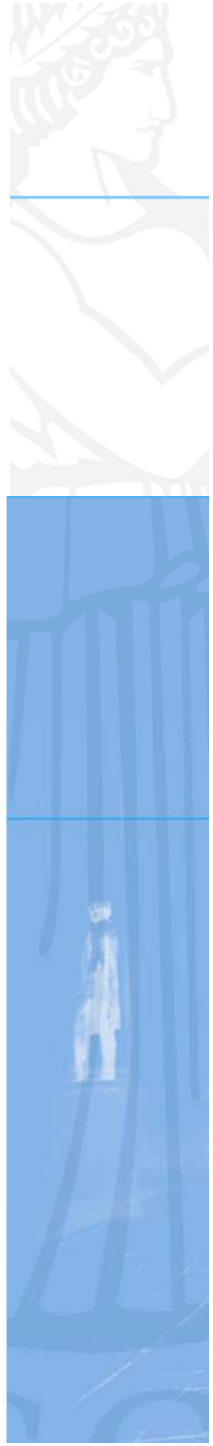
- Litt praktisk informasjon om kurset
- Registrering av oppmøte i pausen
- Noen enkle Java-programer
- Enkel bruk av datamaskinen:
  - Editering
  - Kompilering
  - Kjøring av programmer
- Mål: Du greier å bruke maskinen i løpet av denne eller neste uke og har forsøkt å lage et program

UNIVERSITETET I OSLO

Kursevaluering

- Husk kursevaluering!
  - Både en plikt og en rettighet!
  - Alle kurs ved UiO har det, og det har reell påvirkning
- Inf 1000 har to evalueringer:
  - midtveisevaluering i oktober
  - sluttevaluering i november/desember
- Evaluering av forkurset i informatikk skjer nå:

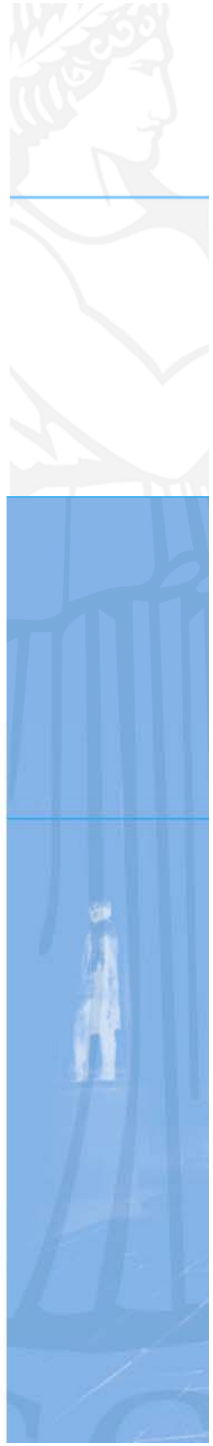
forkurs.ifi.uio.no



UNIVERSITETET I OSLO

Mål for INF1000

- Gi grunnleggende forståelse av noen sentrale begreper, problemstillinger og metoder innen informatikk
- **Lære å programmere**
- Gi noe innsikt i datamaskiners muligheter og begrensninger
- Lære noe om samfunnsmessige konsekvenser av bruk av informasjonsteknologi



UNIVERSITETET I OSLO

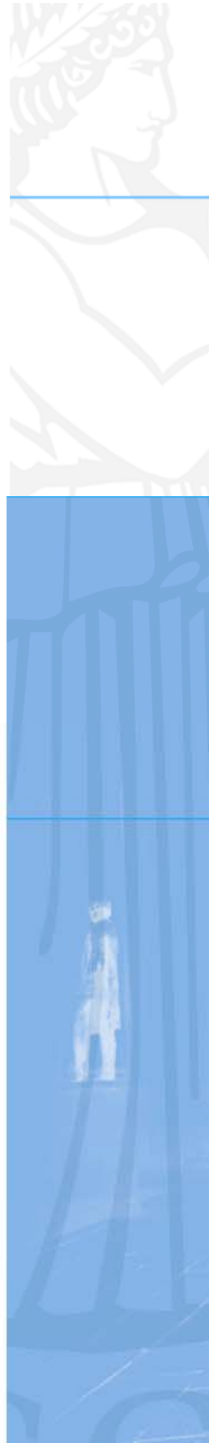
INF1000: oversikt

- Innhold:
  - Litt datateknologi
  - Noe tekstbehandling
  - Mye programmering
- Verktøy:
  - Datamaskiner med Unix og Windows på Blindern eller hjemme-PC med Windows
  - Tekstbehandlingssystemet Emacs på Unix og f.eks Emacs eller TextPad på PC
  - Programmeringsspråket Java
- N.B: Å lære å programmere er mer enn bare å lære et programmeringsspråk

UNIVERSITETET I OSLO

Undervisningen høsten 2010

- Forelesninger
  - 2 timer hver uke
  - Noen er repetisjonstimer og gjennomgang av eksamensoppgaver
- Gruppetimer
  - Oppgavegjennomgang i 2 timer hver uke
    - Gjennomgang av ukeoppgaver
  - Terminaltimer, 2 timer terminal/hjelp i uka .
    - Hjelp til å lese oppgaver praktrisk på terminal – ukeoppgaver og oblig'er
- Selvstudium
  - Lesing, programmering – også løse egne oppgaver, mange timer pr. uke

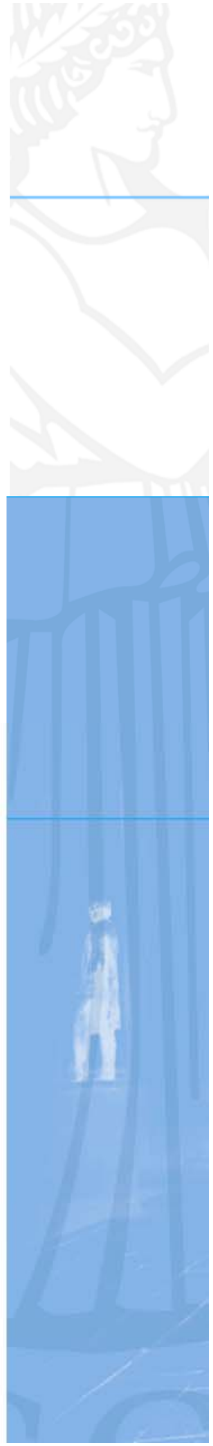


UNIVERSITETET I OSLO

Undervisningsmateriell, del I

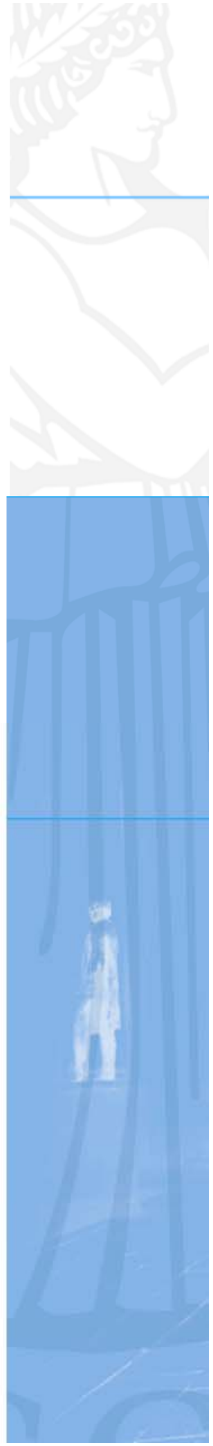
- Lærebok - kjøpes i Akademika:
  - Brunland, Hegna, Lingjærde og Maus: ***Rett på Java,*** *2. utg* (Universitetsforlaget, 2007)
- Følgende lastes snart ned via hjemmesida til kurset :

  http://www.uio.no/studier/emner/matnat/ifi/INF1000/h10/
  - Unix for nybegynnere
    - kompendium av Dag Langmyhr
  - Local guide til Emacs
    - kompendium av Dag Langmyhr
  - Informasjonsteknologi, vitenskap og samfunnsmessige virkninger
    - kompendium av Arne Maus
    - N.B. Vent på oppdatert notat!



Undervisningsmateriell, del II

- Hjemme-DVD for PC med mye nyttig programvare
  - Bla. Emacs, TextPad, Java, Pyton og en rekke andre programmeringsspråk
  - Deles ut gratis
- Kopier av lysarkene fra forelesningene
  - Kan lastes ned over nettet fra hjemmesida og leses på maskin eller skrives ut
  - Dere betaler litt for utskrifter ut over de første 100 ark



UNIVERSITETET I OSLO

Fire oblig’er

- En obligatorisk (programmerings-) øvelse ca. hver tredje uke
  - **Individuell** besvarelse !
  - Leveres hjelpelærer til retting/godkjenning før fristen
  - Hjelp og tips fra medstudenter tillatt, men kopi strengt forbudt (like besvarelser – vil bli sjekket av et program og bli behandlet som fusk)
  - Kan hende du må forbedre ditt løsningsforslag
  - Dere som har tatt inf1000 før: Gamle godkjennelser fortsatt gyldige. Sjekk med gruppelærer
  - ***Alle obligene skal leveres elektronisk – se oppgaveteksten!***
- Første oblig: Innlevering 10. september!

UNIVERSITETET I OSLO

Ukeoppgaver

- Øvelsesoppgaver, nytt sett hver uke
- Flere enn du greier å gjøre
- Gjennomgås på gruppene
- Løsningsforslag gis
- Gå på gruppene - det er der man får kontakt med andre studenter - og man lærer mye av hverandre (og gruppelæreren)
- Gruppeundervisningen starter neste uke!
- Sjekk rom på kurssiden før oppmøte!

UNIVERSITETET I OSLO

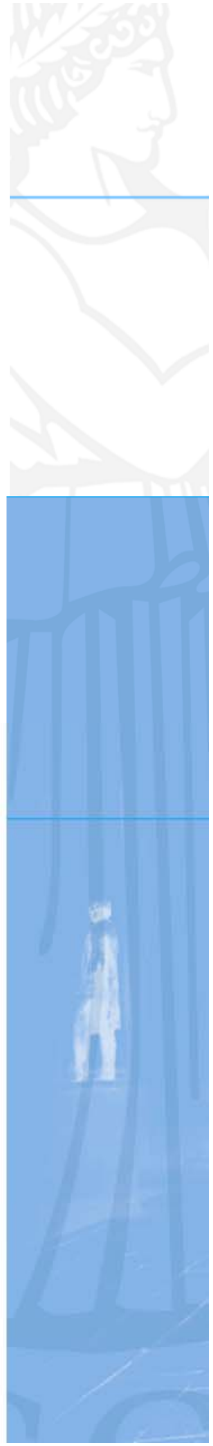
”Grublegruppen”

- En tilbud til de som ønsker mer utfordring
- Start: 16. september
- Påmelding fra 7.9 på epost til martiert@ifi.uio.no

UNIVERSITETET I OSLO

7 “terminal-stuer”

- Abel
  - Størst, men også fullest
- VB
  - nest størst + noen mindre rom
  - Både VB og Abel er helt fulle rett før obliginnlevering!
- Informatikk-bygget:
  - Mindre og bedre plass
- Muligheter for bruk av andre MatNat-maskiner på Bio, Fysikk, Kjemisk
  - Best plass og minst
- Terminalvaktene på Abel, VB og Bio-bygget hjelper deg !
- Både Windows og Linux (Unix) maskiner



UNIVERSITETET I OSLO

Termvakter

- Sjekk åpningstider selv på:
  - http://termvakt.uio.no/wiki/
- VB (betjent)
- Abel (betjent)
- Fysikk (betjent)
- Biologi (betjent)
- PO (ubetjent)
- IFI (ubetjent) (med kort+kode)

UNIVERSITETET I OSLO

Brukernavn og Passord

- For å få adgang til maskinen trenges to opplysninger
  - **Brukernavn** (en kortform av navnet ditt) – dette er offentlig. Foreleseren har f.eks brukernavnet: *arild*
  - **Passord** (hemmelig) – tastes inn etter at du har oppgitt brukernavn. Gir sikkerhet for deg.
- Hvordan få brukernavn og passord ?
  - I posten når du er semesterregistrert (eller av termvakt hvis du ikke finner ditt)
  - Kontoen virker ikke før du har betalt semesteravgift!
  - Trenger du adgang til Ifis maskiner
    - NEI – hvis du aldri er på Blindern og har eget internett-abonnement
    - **JA** – ellers
- Se:http://www.usit.uio.no/it/student/

UNIVERSITETET I OSLO

Eksamen (4 timer)

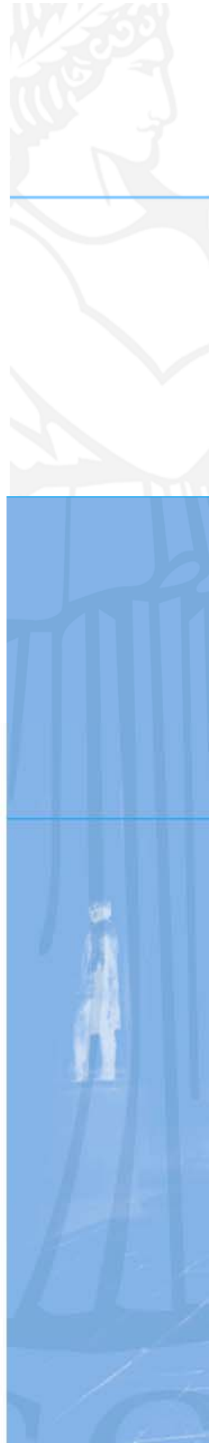
- 1. desember kl. 14:30-18:30
  - Trekkefrist er 1. november
- *Alle* skriftlige hjelpemidler tillatt
- Karakter: A, B,..., E og Stryk (F)
- For å stå i INF1000 må **både** alle de 4 obligene være godkjent **og** eksamen bestås
- Karakteren bestemmes av eksamensbesvarelsen

Hva er et program?

- Maskinen er ganske innskrenket, men kan noen få typer ordre:
  - Les inn et tall (fra tastatur)
  - Skriv ut en tekst (til skjerm, disk,..)
  - Legg sammen to tall
  - ....
- For å få gjort det vi vil, ber vi maskinen utføre et antall slike ordre/handlinger (én etter én)
- Denne rekken av ordre kalles et **program**

Et program minner om en oppskrift

- Vi kjenner andre typer oppskrifter:
  - matoppskrift
  - strikkeoppskrift
  - pianonoter
  - .....
- Et program er en oppskrift til en datamaskin
- Med noter lager man ulike melodier ved å kombinere et mindre antall lyder fra pianoet
- Med programmering kan men lage alle mulige programmer ved å kombinere et begrenset sett av enkle operasjoner i datamaskinen
- Husk: Å *følge* en oppskrift er noe annet enn å *lage* en oppskrift



Hvordan får vi våre programmer inn i maskinen?

N.B.: Det finnes allerede en rekke programmer inne i datamaskinen:

- operativsystemet
- (program-) editoren (emacs, TextPad, WordPad,...)
- oversetteren (kompilatoren)
- kjøre-programmet
- .....

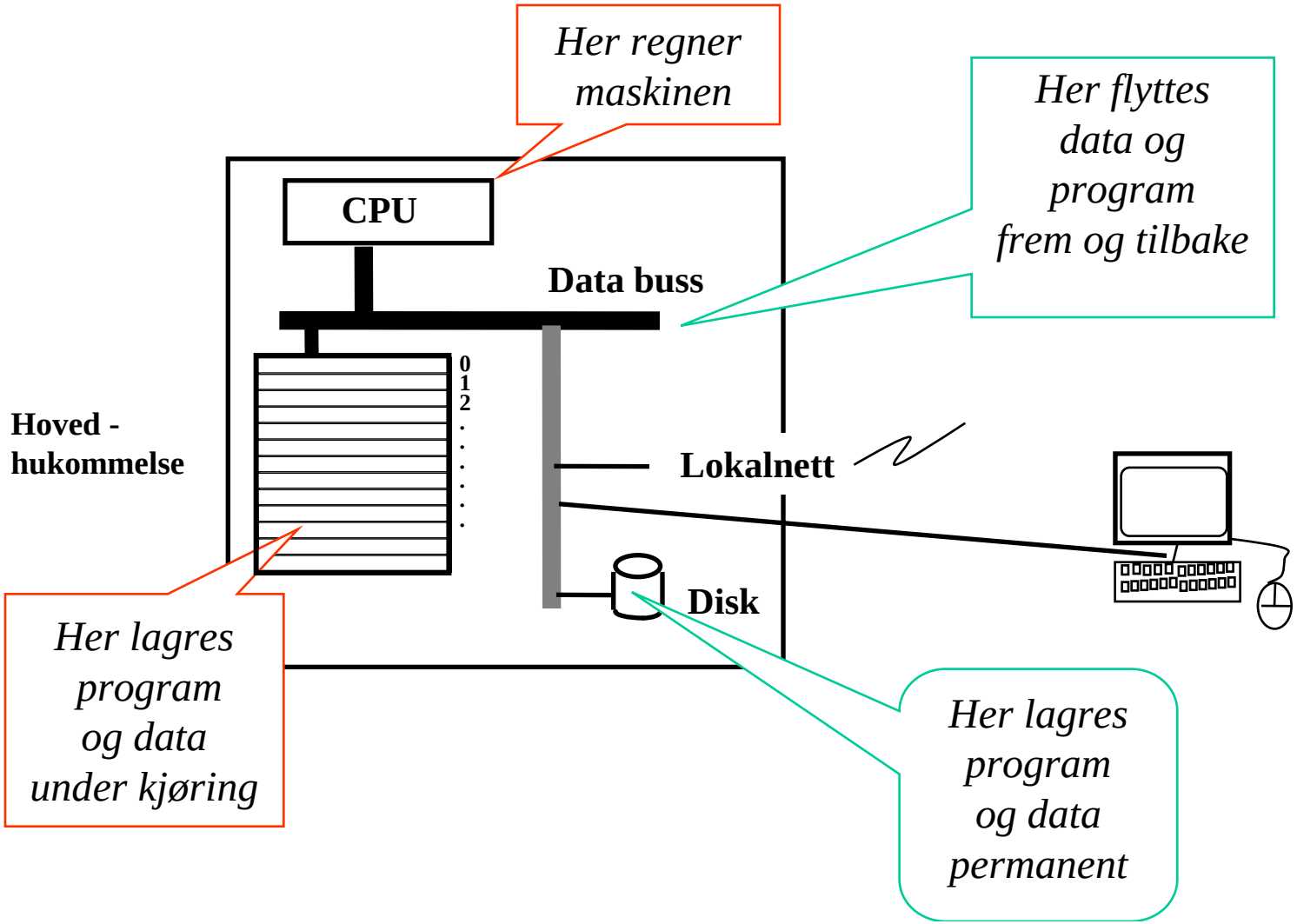
Det er disse programmene som *hjelper* deg til å få ditt program inn i maskinen

Programmering

- Vi skriver våre programmer på en måte som er lettest for oss mennesker (til editoren)
- Denne skrivemåten kalles et programmeringsspråk
- En programtekst skrevet i et slikt programmeringsspråk kan:
  - lett oversettes (av oversetteren) til enkle operasjoner,
  - som lagres i hovedhukommelsen og
  - så kjøres (av kjøre-programmet)
- Det er mange programmeringsspråk - det vi bruker i INF1000 heter **Java**

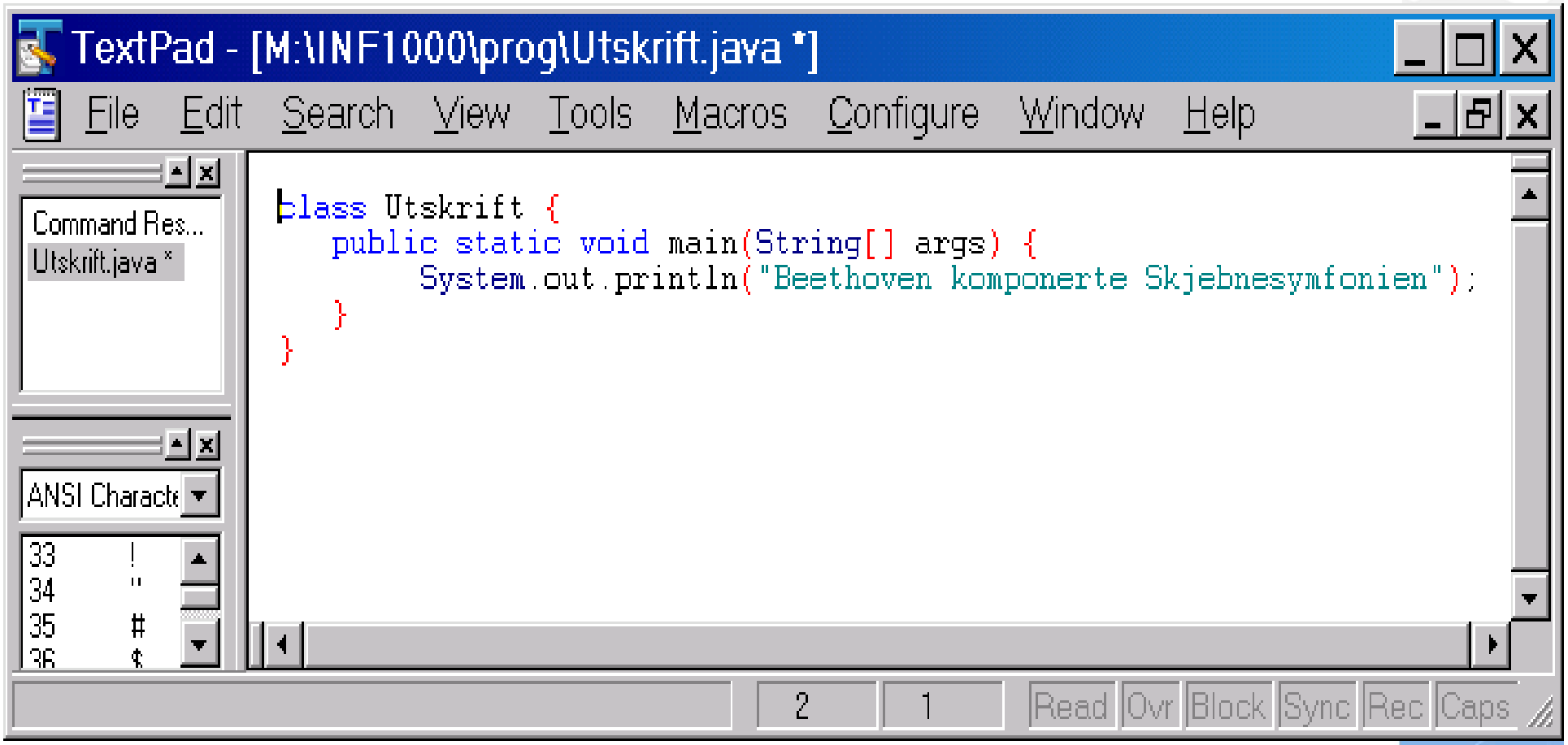
UNIVERSITETET I OSLO

Hva er en datamaskin?



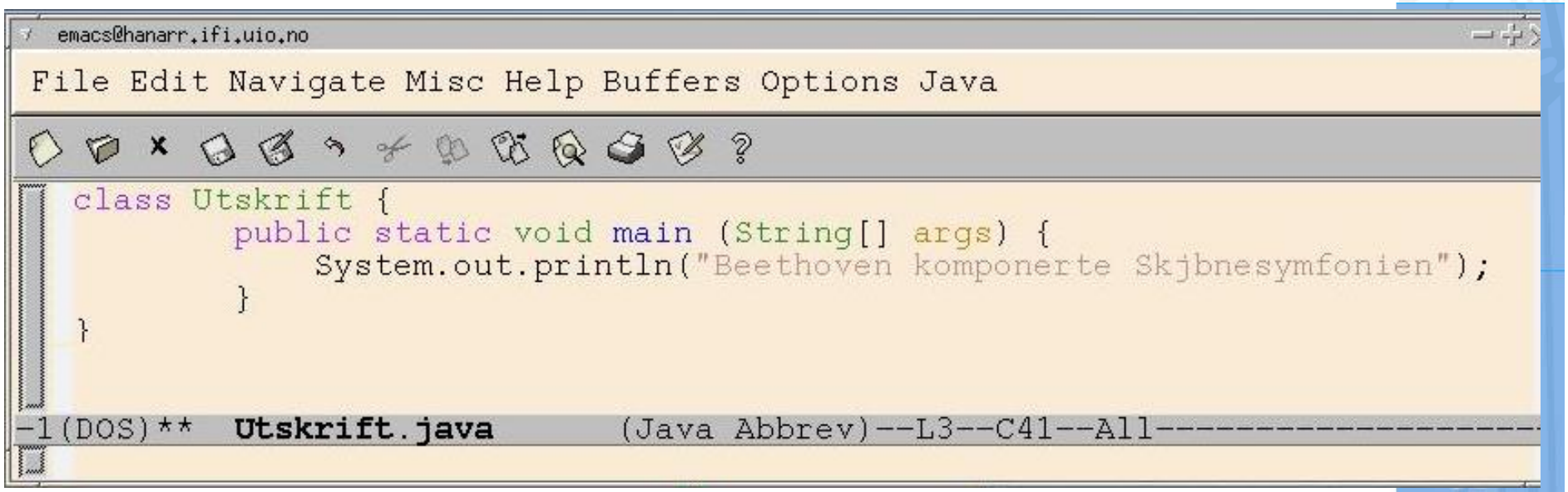
UNIVERSITETET I OSLO

Et første program i Java (– her inne i TextPad)



UNIVERSITETET I OSLO

Et første program i Java (– her inne i emacs)



UNIVERSITETET I OSLO

Vårt første program

```
class Utskrift {
      public static void main(String[] args) {
            System.out.println("Beethoven komponerte Skjebnesymfonien");
      }
}
```

- Et Javaprogram består av minst:
  - En klasse – her: class Utskrift
  - En metode som heter **`main`**
  - Inne i metode main er det én eller flere ordre – her:

```
System.out.println("Beethoven komponerte Skjebnesymfonien");
```

UNIVERSITETET I OSLO

Kompilering (=oversetting) og kjøring (av det oversatte)

```
>javac Utskrift.java
```

Her ber vi om at det oversatte programmet (i Utskrift.class) skal kjøres

```
>java Utskrift
Beethoven komponerte Skjebnesymfonien
```

Denne linja er resultatet av kjøring av programmet

Kompilering: hva skjer?

```
class MittProgram {
    public static void main(String[] args){
        System.out.println("");
    }
}
```

Java Programtekst

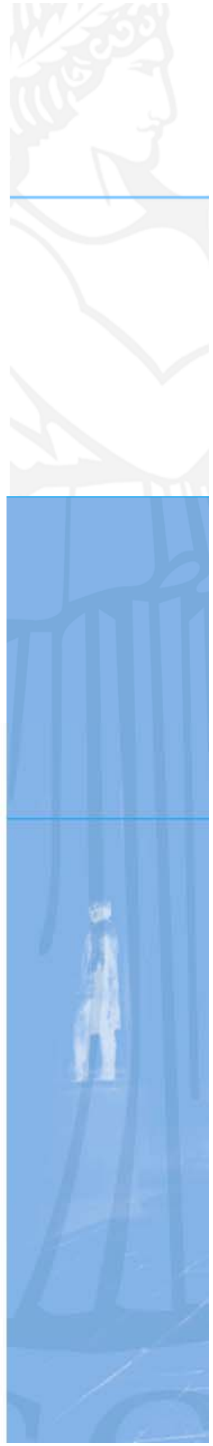
Fil: MittProgram.java

$ javac MittProgram.java

Fil: MittProgram.class

Êþº¾??-?
???
??<init>?()V?Code?LineNumberTable?main?([Ljava/lang/String;)V?
SourceFile?MittProgram.java???
MittProgram?java/lang/Object?
??????????????????????*·?±???????????? ??
??????????<±???????
????????
????

Kompilert Javaprogram



UNIVERSITETET I OSLO

Kjøring : hva skjer?

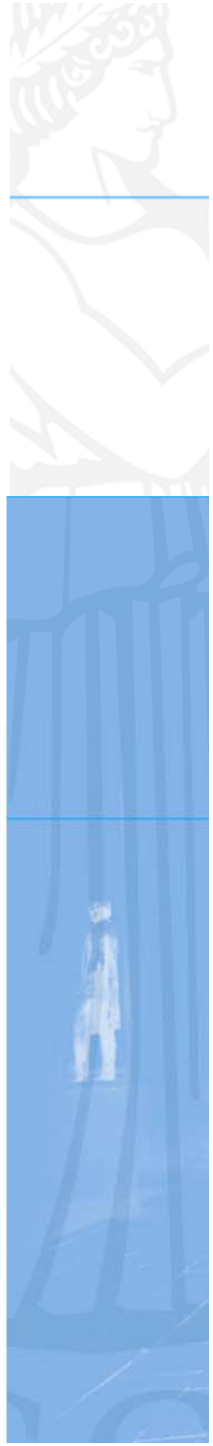
Êþº¾??-?
???
??<init>?()V?Code?LineNumberTable?main?([Ljava/lang/String;)V?
SourceFile?MittProgram.java???
MittProgram?java/lang/Object?
??????????????????????*·?±???????????? ??
??????????<±???????
????????
????

Kompilert Javaprogram

Fil: MittProgram.class

$ java MittProgram

Programmet kjører



Programmering generelt

- Vi skriver programmet som en tekst i en editor
- Vi lagrer filen med programmet lik navnet på klassen og med **`java`** etter punktum – her: **`Utskrift.java`**
- Vi lar kompilatoren **`javac`** oversette **`.java`** filen og legge oversettelsen i en ny fil - her: **`Utskrift.class`**
- Vi starter opp kjøresystemet **`java`** med **`Utskrift`** som parameter på samme linje (den forstår at dette er **`Utskrift.class`**)
- Kjøresystemet leser så denne og utfører de instruksjonene som ligger på **`.class`** fila - her: **`Utskrift.class`**
- Kommandoene som ligger i **`main`** blir da utført,
  - en etter en
  - ovenfra og nedover (til vi har utført siste ordre i main)

UNIVERSITETET I OSLO

Et litt større program – tre linjer utskrift med kompilering og kjøring

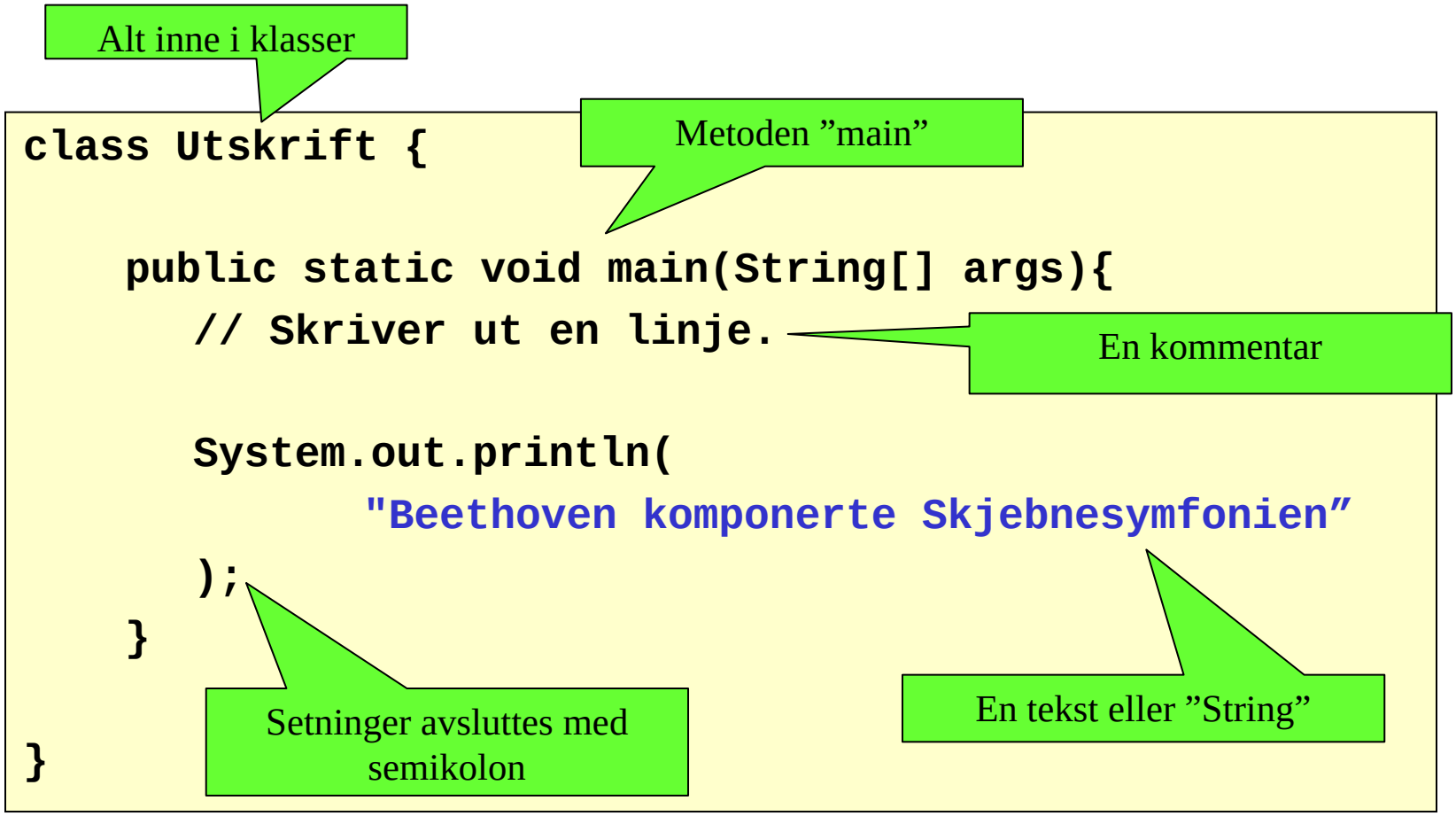
```
class Utskrift2 {
   public static void main(String[] args) {
        System.out.println("Arne har aldri komponert en symfoni");
        System.out.println("Beethoven komponerte Skjebnesymfonien");
        System.out.println(" ------------------******-----------");
   }
}
```

Kompilering og kjøring:

```
>javac Utskrift2.java

>java Utskrift2
Arne har aldri komponert en symfoni
Beethoven komponerte Skjebnesymfonien
 ------------------******-----------
```

UNIVERSITETET I OSLO

Eksempelet igjen – linje for linje




```
class Utskrift {

    public static void main(String[] args){
       // Skriver ut en linje.

       System.out.println(
              "Beethoven komponerte Skjebnesymfonien”
       );
    }
}
```

class Utskrift

- En setning av typen

```
class <klassenavn> {
  <...sekvens av instruksjoner...>
}
```

  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.

`public static void main(String[] args)`

- Må være med i et fullstendig program
- I starten av kurset legger vi all programkode inne i main-metoden
- Senere skal vi lære hva alle ordene betyr!

Kommentarer i programmer

- Kommentarer gjør programmene lettere å forstå
- De oversettes ikke: kompilatoren hopper over dem
- To typer kommentarer:

```
// Her er en kommentar som varer ut linja

/* Her er en kommentar som varer
        helt til hit */
```

- Gode programmer har kommentarer, men ikke på hver linje!
- Dere må kommentere programmene til oblig 2-4!

Nytt eksempel: Gangetabell

```
class Gangetabell {
   public static void main (String [] args) {
      System.out.println(1 * 8);
      System.out.println(2 * 8);
      System.out.println(3 * 8);
      System.out.println(4 * 8);
      System.out.println(5 * 8);
   }
}
```

KOMPILERING OG KJØRING

```
> javac Gangetabell.java
> java Gangetabell
8
16
24
32
40
```

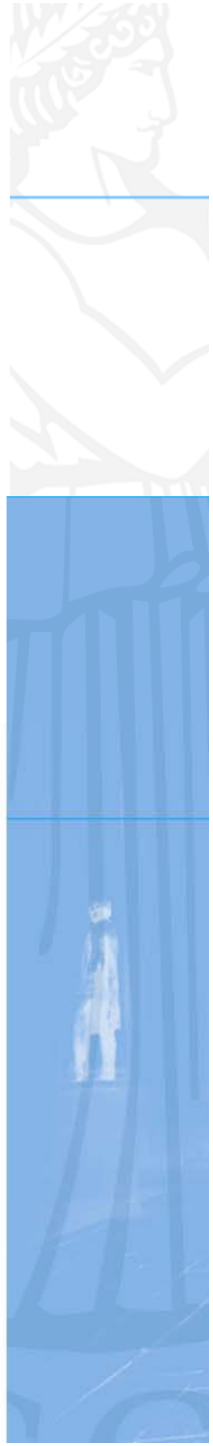
Variable – Programmer og data

- Programmet bearbeider data
- Oppgaver som søk, sortering, beregning
  - Summere regning
  - Finne studenten med best gjennomsnittskarakter
  - Finne billigste flybillett
  - Regne ut hvordan været blir i morgen
- Vi må sette av plass til dataene

Variabel – En plass i lageret

- En plass i maskinens lager (minne) ligner
  - en skuff i en kommode
  - en biloppstillingsplass på en parkeringsplass
- De kan ha forskjellige størrelse avhengig hvilke dataelementer som skal lagres der

- Variable må ha **navn**
  - Slik at vi kan referere til dem
- Variable må ha **type**
  - Så vi vet hvordan data som kan lagres



UNIVERSITETET I OSLO

Hvordan deklarerer vi variable?

- Deklarasjon angir navn og type til en variabel
- Vi deklarerer en variabel bare én gang

- Eksempel: En heltallsvariabel kan deklareres med typen int (for ”integer”):

```
int tall;
```

En setning avsluttes alltid med semikolon

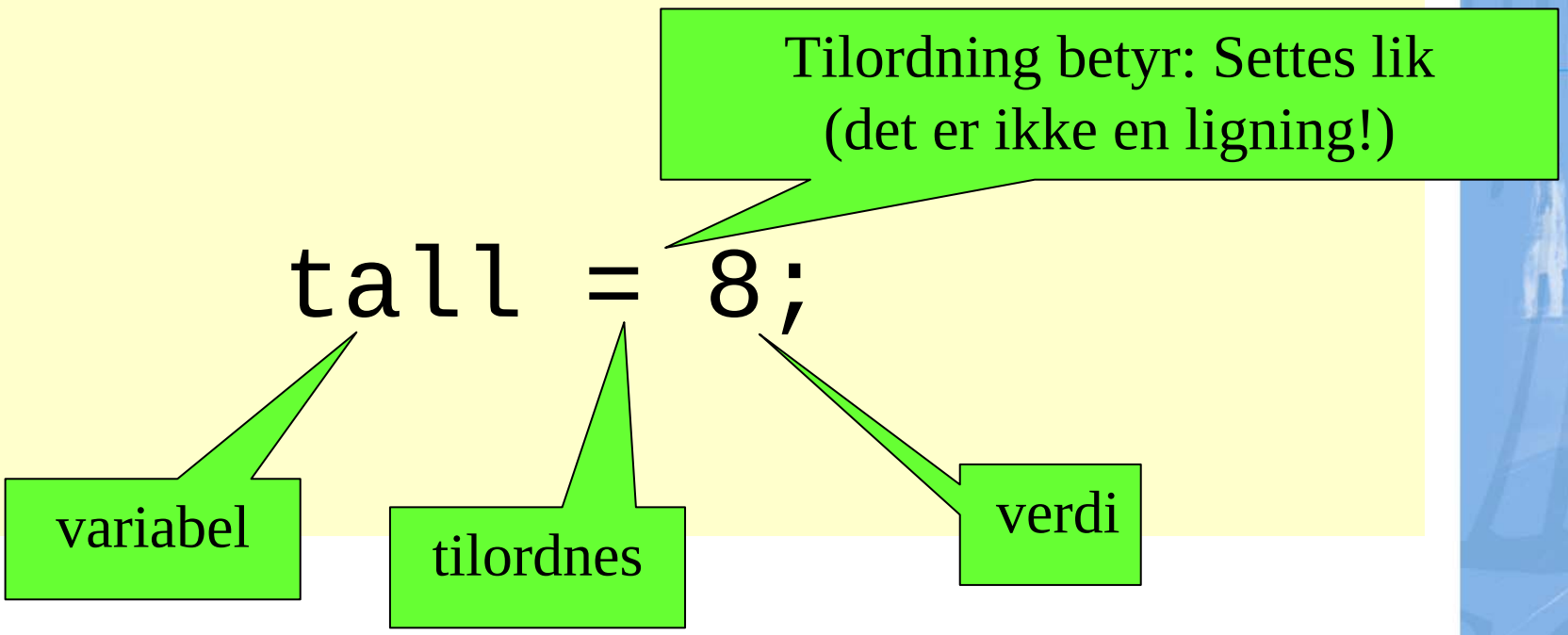
Type

Navn

UNIVERSITETET I OSLO

Tilordning av verdi til variable

- En variabel har ingen verdi i utgangspunktet
- Den kan gis verdi med en *tilordningssetning*
- En deklarert variabel kan gis verdi flere ganger



Eksempel med variable

```
class VariabelEksempel {
   public static void main (String [] args) {
      int tall;

      tall = 8;
      System.out.println(tall);
      tall = 2 * tall;
      System.out.println(tall);
   }
}
```

KOMPILERING OG KJØRING

```
> javac VariabelEksempel.java
> java VariabelEksempel
8
16
```

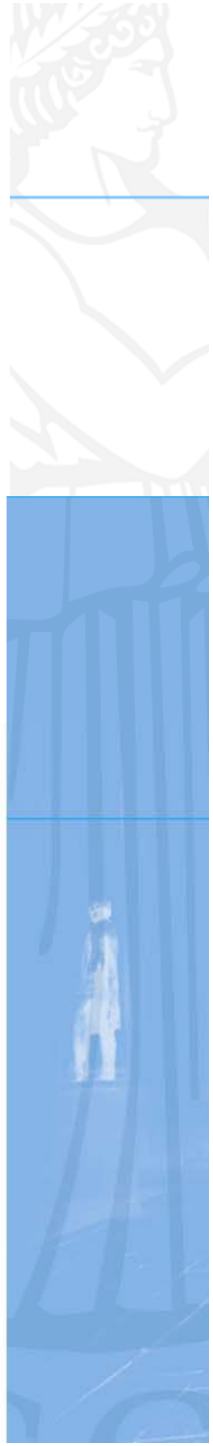
Tilordningen – hva skjedde der?

```
tall  = 2 * tall;
```

1. Verdien som ligger i variabelen tall hentes fram (her: 8)
2. En ny verdi beregnes ut fra uttrykket ”2 * tall” (her: 16)
3. Variabelen tall settes til denne nye verdien

Variabelen tall har med andre ord verdien:

- 8 før setningen er utført
- 16 etter den er utført



UNIVERSITETET I OSLO

Vi må gi verdi før vi bruker den

- En variabel som ikke er tilordnet kan ikke avleses
- Gir feil når vi forsøker å kompilere programmet

```
int tall;
tall = 2 * tall;
```

Forsøker å lese en variabel som ikke er tilordnet

```
$ javac TilordningAvlesing.java
```

Kompilerer programmet

```
TilordningAvlesing.java:7: variable tall might not have been initialized
        tall = 2 * tall;
               ^
1 error
```

Vi får en feilmelding

Gangetabell med variable

```
class GangetabellVar {
   public static void main (String [] args) {
      int tall;
      tall = 8;
      System.out.println(1*tall);
      System.out.println(2*tall);
      System.out.println(3*tall);
      System.out.println(4*tall);
      System.out.println(5*tall);
   }
}
```

KOMPILERING OG KJØRING

```
> javac GangetabellVar.java
> java GangetabellVar
8
16
24
32
40
```

UNIVERSITETET I OSLO

Variable – Flere i samme setning

```
int lengde, bredde, høyde;
```

er det samme som

```
int lengde;
int bredde;
int høyde;
```

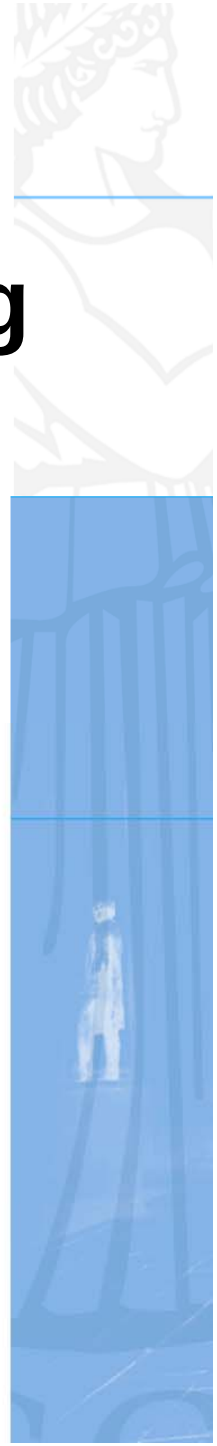
UNIVERSITETET I OSLO

Kombinert deklarasjon og tilordning

```
int tall = 8;
```

er det samme som

```
int tall;

tall = 8;
```



UNIVERSITETET I OSLO

Bytte verdier mellom to variable

- Anta at vi har disse instruksjonene:

```
int første, andre;
første = 65;
andre = 77;
```

- Hvordan kan vi bytte om verdiene i de to variablene?
- Vi forsøker dette:

```
første = andre;
andre = første;
```

- Hvorfor virker ikke dette?

Når vi har utført …	så er verdien til:
`første = 65;` `andre = 77;`	**første: 65** **andre : 77**
`første = andre;`	**første: 77** **andre : 77**
`andre = første;`	**første: 77** **andre : 77**

UNIVERSITETET I OSLO

Løsning: hjelpevariabel

- Vi tar vare på den opprinnelige verdien i en tredje variabel:

```
int første, andre, minne;
første = 65;
andre = 77;
minne = første;
første = andre;
andre = minne;
```

- Vi sjekker at det virker:

Når vi har utført …	så er verdien til:

`første = 65;`
`andre = 77;`
	`første:  65`
	`andre :  77`
	`minne :  --`

`minne = første;`
`første = andre;`
	`første:  77`
	`andre :  77`
	`minne :  65`

`andre = minne ;`	`første:  77`
	`andre :  65`
	`minne :  65`

Heltall og desimaltall

- To viktige datatyper:
  - int: heltall
  - double: desimaltall (flyttall)
- Ved tilordning må typen til verdien være den samme som typen til variabelen:

```
int verdi1 = 12;          // OK
int verdi2 = 2.0;         // Gir kompileringsfeil

double verdi3 = 2.0;      // OK
double verdi4 = verdi1;   // OK - gjør om til 12.0
```

Typekonvertering

- Det er mulig å konvertere fra en datatype til en annen
- Enkelt fra heltall til et flyttall.
- Den andre veien må vi informere kompilatoren om
- Vi gjør det ved å sette typenavnet i parentes rett foran verdien vi ønsker å konvertere

```
double d = 3.14;
int i = (int) d;
int j = (int) 2.222;

// Men dette er altså ok
int x = 9;
double db = x;
```

UNIVERSITETET I OSLO

Hvorfor ikke alltid bruke double?

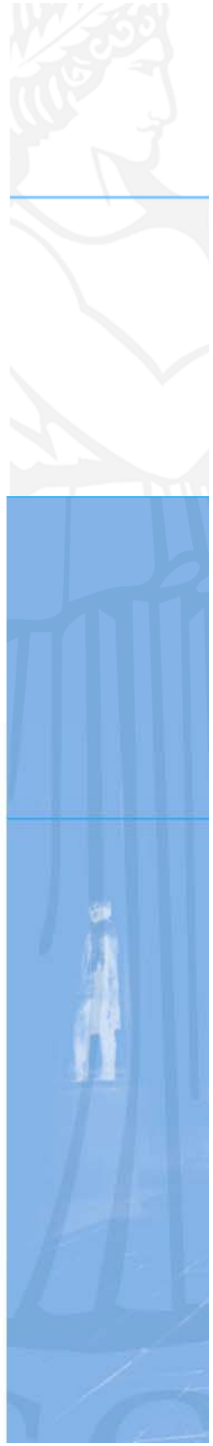
- Mens regning med heltall alltid er eksakt, er regning med desimaltall ikke:

```
double x = 0.1;

double y = (x + 1) - 1;

// Nå har ikke x og y samme verdi!
```

- x og y er nesten like, men det er forskjell i et av desimalene langt ute
- Når det er naturlig å bruke heltall bruker vi int!
- Når det er naturlig å bruke desimaltall bruker vi double

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.



Kompileringsfeil

```
class FeilType {
       public static void main(String[ ] args){
              double d = 1.5;
              int i;
              i = d;
       }
}
```

```
$ javac FeilType.java
FeilType.java:5: incompatible types
found   : double
required: int
         i = d;
             ^
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)
```

Tekst i programmer

- Datatypen for tekst heter String:

```
String tekstSnutt = ”Dette er en tekst”;
```

- String-verdier kan settes sammen med +:

```
String tillegg = ” som vises på forelesning”;
String fulltekst = tekstSnutt + tillegg;
System.out.println(fulltekst);
```

Dette er en tekst som vises på forelesning

UNIVERSITETET I OSLO

Datatypen avgjør hva ”+” betyr

```
System.out.println(”2” + ”3”);
// Resultat: 23
System.out.println(”2” + 3);
// Resultat: 23
System.out.println(”2 + 3”);
// Resultat: 2 + 3
System.out.println(2 + 3);
// Resultat: 5
```

Når + brukes mellom strenger, betyr det streng-sammenslåing

Når et argument til + er en streng, vil de andre konverteres til strenger

Eksempel: Regne ut areal

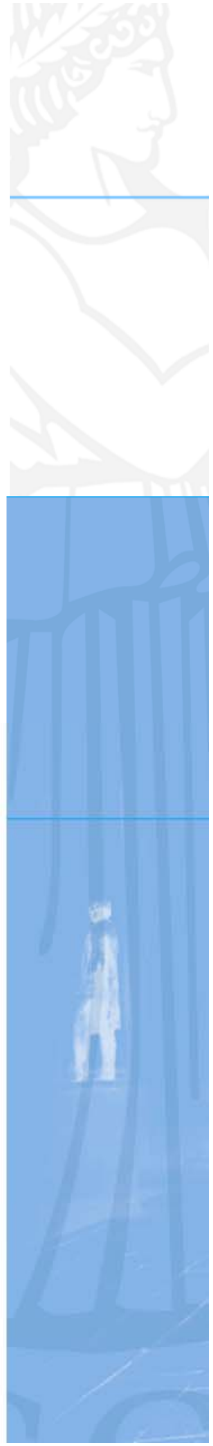
```
class Repetisjon {
   public static void main(String[] args){
       final double PI = 3.14;
       double radius = 2.0;
       double areal;
       String fortekst =
             "Arealet til en sirkel med radius ";

       areal = PI * radius * radius;

       System.out.println(fortekst + radius +
             " er " + areal + ".");
   }
}
```

final betyr konstant

Arealet til en sirkel med radius 2.0 er 12.56.

Hvordan løse oppgaver

1. **Se oppgaven 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. **Hvordan transformere inndata til utdata?**
    1. Hvordan skal representeres (lagres)?
    2. Spesifiser en sekvens av trinn der:
        - hvert trinn gjør en enkel ting med dataene
        - 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

UNIVERSITETET I OSLO

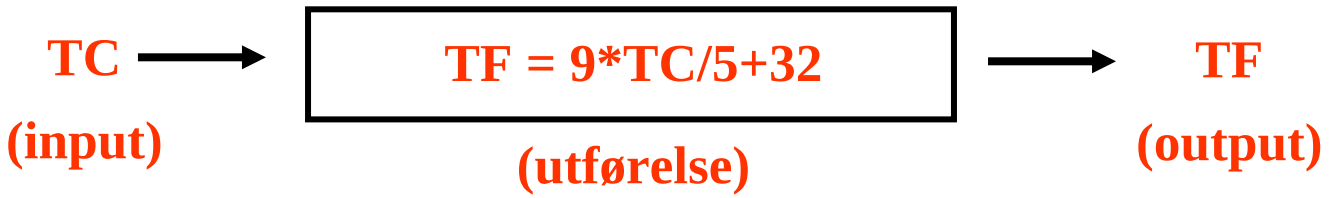
Transformere inndata til utdata

- 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 tempCelcius, tempFahrenheit;
        System.out.println("Celcius Fahrenheit");

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

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

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

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

```
Celcius Fahrenheit
-10.0 14.0
0.0 32.0
37.0 98.6
100.0 212.0
```