INF 1000 – høsten 2011
Uke 3 – 6. september

Grunnkurs i Objektorientert Programmering
Institutt for Informatikk
Universitetet i Oslo

Siri Moe Jensen og Arne Maus

Innhold – uke 3

- Rep: variable og uttrykk
- Konvertering
- Å skrive et program
- Arrayer
- while-løkke, do-while, for-løkke
- Tastatur/ skjerm input/ output - easyIO

Mål for uke 3:

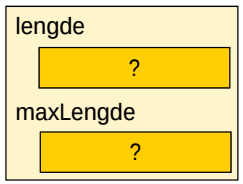
* Java: Arrayer, løkker, lese fra tastatur/ skrive til skjerm (Kap 3.1-3.5, 4.3-4.4, 5.1-5.4)

* Programmering: Tenke ut, skrive, teste og rette programmer m/ nye Java-elementer

Variabel: En plass i hukommelsen

- For å kunne huske og forandre på data, trenger programmet *variable* å lagre dem i.
- Vi reserverer plass i hukommelsen ved å *deklarere* en variabel – da oppgir vi *type* og *navn*

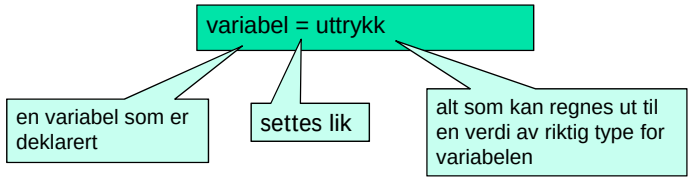
```
int lengde;
int maxLengde;
```



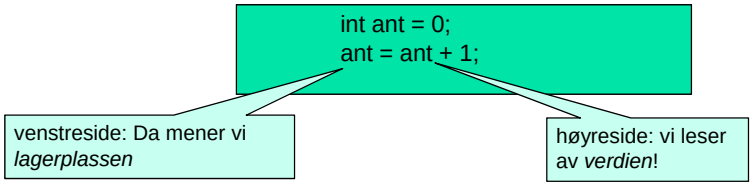
- Typen avgjør *hva slags verdi* som kan lagres i variabelen: int er en type for å lagre heltall
- Du bestemmer navnet: start med (helst liten) bokstav, deretter tall og bokstaver

Uttrykk

- Den generelle formen for en tilordning er



- *Uttrykket* på høyre side leses av og beregnes først – deretter plasseres resultatet som verdi i variabelen på venstre side



Uttrykk og presedens

- De fleste uttrykk er korte og enkle

```
int a = b;
String s = t + u;
```

- Dersom det er den minste tvil, bruk parenteser

```
double d = y + (2.0 * z);
```

- Dere vil gjøre det bra på eksamen selv om dere ikke kan hver minste detalj om uttrykk og presedens (eller konvertering).

Konvertering

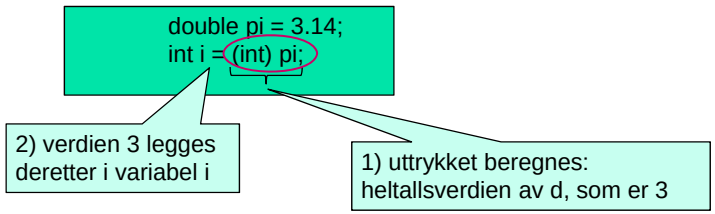
- Vi så sist at numeriske beregninger ga heltall-svar hvis alle verdiene som inngikk var heltall, ellers double-svar
- Tilsvarende ved sammenligning på tvers av numeriske typer
  - Hvis det er både int og double variable i et uttrykk, gjøres de automatisk (implisitt) om til double før uttrykket beregnes.
  - NB dette endrer ikke variable i uttrykket!

```
double pi = 3.14;
int i = 3;

if (i == pi) {}        // vurderes som (3.0 == 3.14) altså false
```


Konvertering ved tilordning

- Tilordning på tvers av numeriske typer
- Plassere int-verdi i en double-variabel: Helt greit (konverteres implisitt – dvs uten at du ber om det)
- Plassere double-verdi i en int-variabel: Må konverteres først (= si fra at du er villig til å miste noe informasjon)
- Kalles eksplisitt konvertering – fordi du ber om det



- NB: variabelen pi er ikke endret etter dette

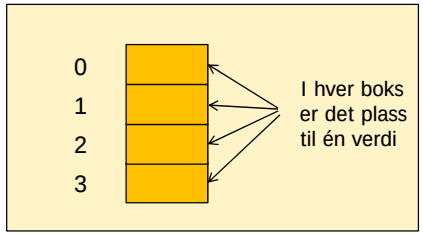
Løsing av oppgaver

Hva er det egentlig vi gjør når vi skriver et program?

1. Hvordan ville jeg selv løse dette problemet?
2. Gitt en datamaskins begrensninger, hvordan må den gå frem for å løse det?
   - trenger navngitte plasser i hukommelsen (datastruktur)
   - trenger en detaljert skritt-for-skritt oppskrift (algoritme)
3. Formulere dette i Java

Arrayer

- En variabel holder kun én verdi
- En *array* kan holde mange verdier av samme type
- Verdiene i en array med lengde N får hver sin indeks (=posisjon) fra 0 til N-1
- En array med lengde 4 kan tegnes slik:



Deklarere og opprette en array

- Deklarere en array (angi type og gi den navn)

F eks int eller char

```
dataType [] arrayNavn;
```

- Opprette en array (sette av plass i hukommelsen)

```
arrayNavn = new dataType [N]
```

N er ønsket lengde

- Deklarere og opprette samtidig:

```
datatype[] arrayNavn = new dataType [N];
```

- Eks:

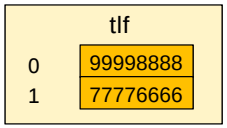
```
int [] soltimer = new int [31];
int lengde = 31;
double [] nedbør = new double [lengde];
String [] navn = new String [5];
```

lengden kan være en variabel

Bruk av arrayen

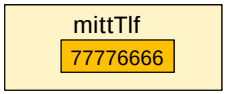
- Plasser i en array brukes som variable: Oppgi arraynavn og indeks
- Tilordning

```
int [] tlf = new int [2];
tlf [0] = 99998888;
tlf [1] = 77776666;
```



- Lesing

```
int mittTlf = tlf [1];
```



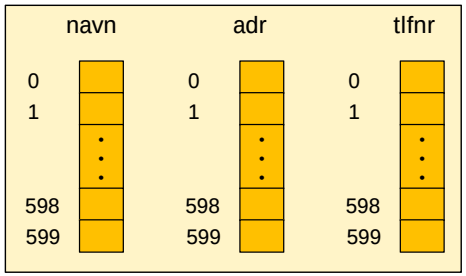
- Få tak i lengden på arrayen

```
int antTlf = tlf.length;
```

Eksempel - arrayer

- Anta at vi ønsker å lagre navn, adresse og telefonnummer for de som følger et bestemt kurs med maksimalt 600 studenter

```
String [] navn = new String [600];
String [] adr = new String [600];
int [] tlfnr = new int [600];
```



Løkker

- Ofte ønsker vi å utføre en eller flere programsetninger mer enn én gang
- Dette gjøres ved hjelp av *løkker*

- Hovedsakelig to typer behov:
  - Utfør setningen(e) inntil en betingelse oppfylles -> while-løkke
  - Utfør setningen(e) et bestemt antall ganger –> for-løkke

while-løkker

- Vi kan utføre en blokk med setninger flere ganger ved hjelp av en while-løkke

```
while (<logisk uttrykk>) {
    <setning 1;>
    <setning 2;>
    .....
    <setning n;>
}
```

- Hvis det logiske uttrykket er true, utføres blokken i while-løkka
- Når blokken som hører til løkka er utført, vil programmet returnere til toppen og evaluere det logiske uttrykket på nytt
- Hvis det logiske uttrykket er false, fortsetter utførelsen etter blokken.

Eksempel

```
class SkrivLinjer {
    public static void main (String [] args) {
        int k = 1;
        while (k <= 5) {
            System.out.println(
                "Nå har k verdien " + k);
            k = k + 1;
        }
        System.out.println("Nå er k lik " + k);
    }
}
```


Kjøring

```
> java SkrivLinjer
Nå har k verdien 1
Nå har k verdien 2
Nå har k verdien 3
Nå har k verdien 4
Nå har k verdien 5
Nå er k lik 6
>
```

Evig løkke

- Dersom testen i while-løkka **aldri blir usann** (false), vil utførelsen av while-løkka aldri stoppe. Dette kalles en evig løkke.
- To eksempler:

```
class EvigLokkeOpplagt {
    public static void main (String [] args) {
        while (true) {
            System.out.println("INF 1000");
        }
    }
}
```

```
class EvigLokkeIkkeSaOpplagt {
    public static void main (String [] args) {
        int i = 1, j = 2;
        while (i < j) {
            System.out.println("Nå er i < j (i=" +
                  i + ", j=" + j + ")");
            i++; j++;
        }
    }
}
```


Evig løkke - Kjøring

Den kan stoppes med **Ctrl+C**

```
> java EvigLokkeOpplagt
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000

. . .
. . .
```

```
> java EvigLokkeIkkeSaOpplagt

. . .
. . .

Nå er i < j (i=82155, j=82156)
Nå er i < j (i=82156, j=82157)
Nå er i < j (i=82157, j=82158)
Nå er i < j (i=82158, j=82159)
Nå er i < j (i=82159, j=82160)
Nå er i < j (i=82160, j=82161)
Nå er i < j (i=82161, j=82162)
Nå er i < j (i=82162, j=82163)
Nå er i < j (i=82163, j=82164)

. . .
```


Variant av while – do-while

- Formen på en do-while løkke:

```
do {
      <setning 1;>
      <setning 2;>
      .....
      <setning n;>
} while (<logisk uttrykk>);
```

- Noen foretrekker denne fremfor while-løkker *når løkke-innmaten alltid skal utføres minst en gang*.

for-løkker

- En annen måte å få utført en instruksjon (eller blokk) mange ganger er ved hjelp av en **for-løkke**:

```
for (<initialisering>; <betingelse>; <oppdatering>){
      <setning 1;>
      <setning 2;>
      ....
      <setning n;>
}
```

Eksempel på for-løkke

```
class ForLokkeHvordan {
    public static void main (String [] args) {
        for( int i=0; i<5; i++) {
            System.out.println("Nå er i= " + i);
        }
}
```

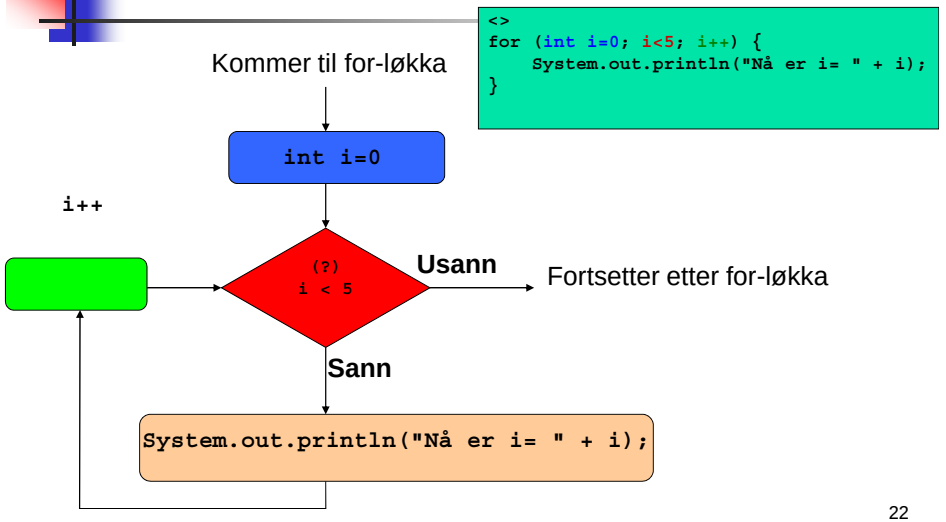
Initialisering

Oppdatering

Betingelse

```
> java ForLokkeHvordan
Nå er i= 0
Nå er i= 1
Nå er i= 2
Nå er i= 3
Nå er i= 4
>
```


Flytdiagram: Hvordan for-løkka virker



Nesting av løkker

- Det er ofte behov for å neste løkke-setninger inne i hverandre
- Man må passe på å bruke forskjellig ”tellevariabel” i hver løkke
- I dette eksemplet er **i** og **j** brukt.

```
class NestetForLokke {
    public static void main (String [] args) {
        for( int i=1; i<=10; i++) {
            for( int j=1; j<=10; j++) {
                int produkt = i * j;
                System.out.println(i + "*" + j +"=" + produkt);
            }
        }
    }
}
```


Nesting av løkker – variant

- Nedenfor har vi endret den innerste for-løkken slik at den ikke skifter linje for hver produkt, men kun når i inkrementeres
- Senere vil vi se hvordan tabellen kan lages med rette kolonner

```
class NestetForLokke {
    public static void main (String [] args) {
        for( int i=1; i<=10; i++) {
            for( int j=1; j<=10; j++) {
                int produkt = i * j;
                System.out.print(i + "*" + j +"=" + produkt);
            }
            System.out.println ();
        }
    }
}
```

Initialisering og utskrift av array

- Vi ønsker å lagre navnet på ukedagene – og bruker en med 7 plasser av typen String
- Deretter skriver vi disse ut på en linje med blanke imellom

```
class NestetForLokke {
    public static void main (String [] args){
        String [] ukedag =
{"mandag","tirsdag","onsdag","torsdag","fredag","lørdag","søndag"};

        for (int i = 0; i<7; i++) {
            System.out.print (ukedag[i] + "   ");
        }
        System.out.println ();
    }
}
```


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 – innlesning med sjekk

- Lag et program som leser et heltall mellom 1 og 100 fra terminal.
- Hvis det innleste tallet ikke ligger i det lovlige intervallet, skal programmet be om nytt tall.
- Dette gjentas inntil brukeren skriver et lovlig tall.
- Skriv til slutt ut en tekst som inneholder tallet.

Program – innlesning med sjekk

```
import easyIO.*;
class LesVerdiSjekk {
    public static void main (String[] args) {
        In tast = new In();
        System.out.print("Oppgi verdi (1,2,...,100): ");

        int verdi = tast.inInt();

        while ((verdi < 1) || (verdi > 100)) {
            System.out.println("Ulovlig verdi!");
            System.out.print("Prøv igjen: ");
            verdi = tast.inInt();
        }

        System.out.println("Du oppga verdien " +
            verdi);
    }
}
```

Kompilering og kjøring

```
> javac LesVerdiSjekk.java
> java LesVerdiSjekk
Oppgi verdi (1,2,...,100): 101
Ulovlig verdi!
Prøv igjen: 0
Ulovlig verdi!
Prøv igjen: 3
Du oppga verdien 3
>
```


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 - evt neste linje hvis tomt:
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 resten av linjen (gjør et linjeskift først om tomt): inLine()
- Lese linje for linje:
  - Kun inneværende linje uansett om den er tom: Bruk readLine()
- Lese tegn for tegn:
  - For å lese neste tegn (også blanke): 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 første linje 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. I INF1000 bruker vi pakken easyIO – samme som for innlesing.
- I toppen av programmet (før class) skriv:

```
import easyIO.*;
```

- Inne i klassen skriver vi så:

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

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

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


Eksempel

Vi må først importere pakken easyIO.

```
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 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.

Resultat

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


Tre måter å skrive ut på

- Uten formatering:

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

- Angi utskriftsbredde:

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

- Angi utskriftsbredde og justering:

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

Hva kan jeg spørre om..?

- Faglige spørsmål, hjelp med oppgaver inkl oblig
  - Gruppelærer og INF1000-blogg (på kurssidene)
  - Studieorakler og termvakter – og medstudenter!
- Installasjon/ oppsett av egen laptop for Ifi-kurs
  - Helpdesk i 3. etg OJD
- Spørsmål om innlevering, gruppedeltakelse
  - Gruppelærer, blogg
- Praktiske spørsmål, krav og regler for programmer og emner
  - Studieadministrasjonen
- Vanskelig å henge med, usikker på studiet, annet
  - studielaben@ifi.uio.no
  - Kollokvierom v/ sonen: Tirsdag 10.30-11.30 + 14.30-15.30