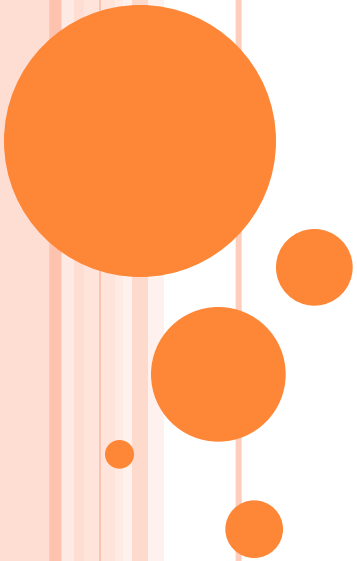


INF1000 Ekstratilbud

Stoff fra uke 1-5 (6)

3. oktober 2012

Siri Moe Jensen

Plan for dagen

- gjennomgå stoff fra uke 1-5(6), men med en litt annen tilnærming
- kun gjennomgått stoff, men vekt på konsepter og forståelse
- => hva er det egentlig som skjer (på et ”passe” høyt nivå)
- bruke tid på å tegne datastrukturer, vise hva som skjer med variable og hvordan setningene i programmet utføres

Hva lærer dere av Java i INF1000?

- Hukommelse/ datastruktur
  - enkle variable
  - arrayreferanse og –objekter (se lysark fra uke 5)
  - tekstreferanse og –objekter
  - i/o buffere
  - blokker
  - (klasser og objekter)
- Programutførelse/ kontroll
  - if
  - while, for
  - metode-kall
  - konstruktører

Deklarasjon av enkle variable

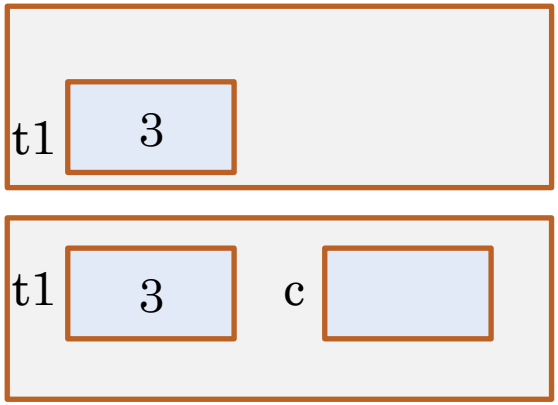
- eksempler: char, int, boolean, double
- Kan deklareres uten initiell verdi, eller deklareres med initialisering (får da et verdi med det samme, og kan leses av uten feilmelding)

Java-setning

```
int  t1 = 3;

char c;
```

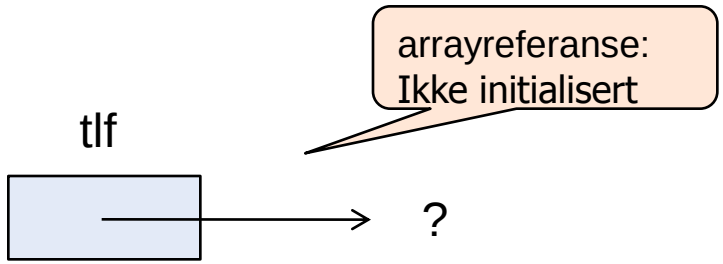
Hukommelsen



Deklarasjon av peker og opprettelse av arrayobjektet

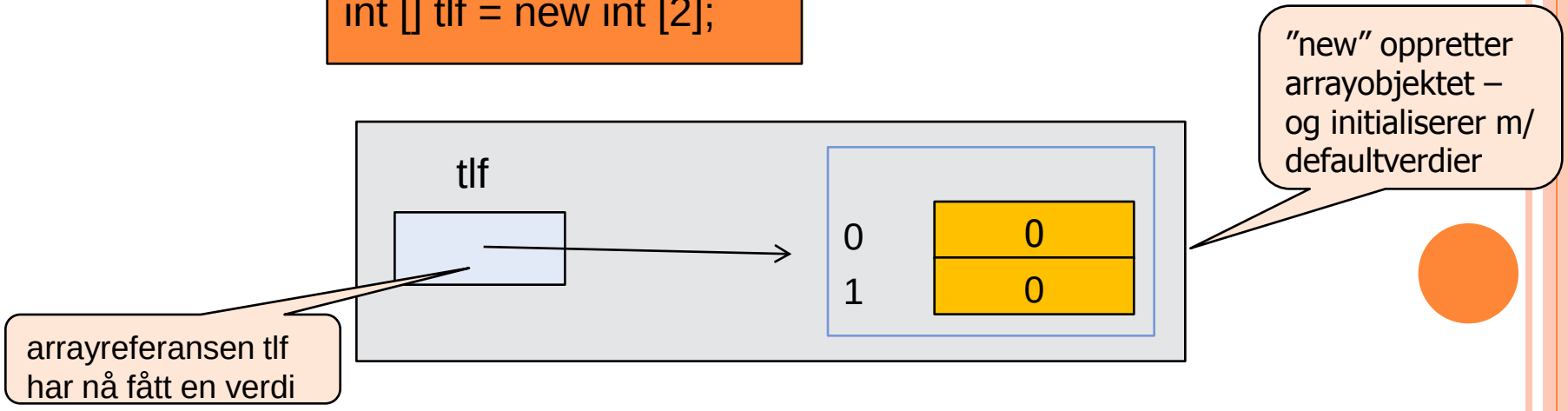
- Deklarasjon av arraypeker

```
int [] tlf;
```



- Arrayobjekt med angitt størrelse opprettes – og pekes på av tlf

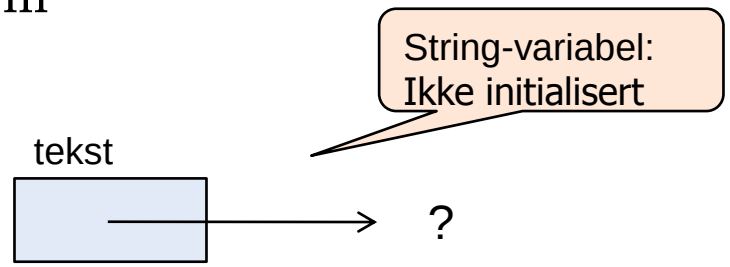
```
int [] tlf = new int [2];
```



Tekster - String

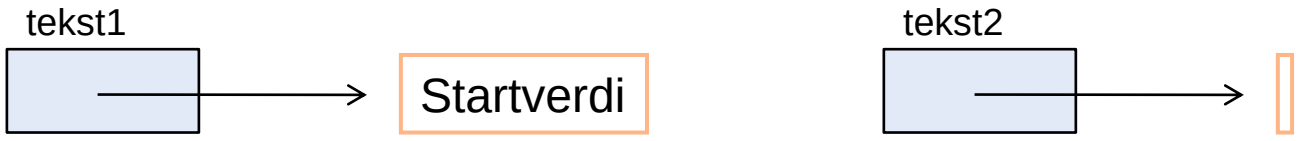
- En tekst er en sekvens av tegn
- I Java deklareres tekstvariable som

```
String tekst;
```

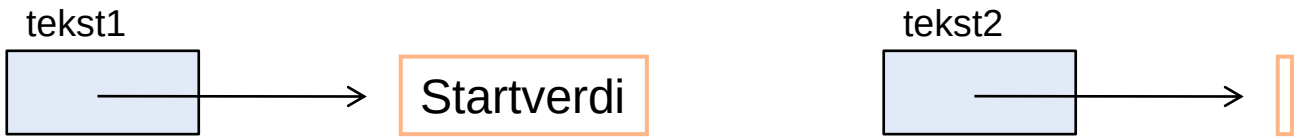


- Vi kan gi en initiell verdi, for eksempel

```
String tekst1 = ”Startverdi”;
String tekst2 = ””;                    // en tom streng
```



Litt mer om String-klassen



- String er en klasse, med en rekke metoder.
- Disse metodene kan kalles for en gitt String-variabel (tekstpeker), f eks:
  - **tekst2.length();** //returnerer verdien **0**
  - **tekst1.charAt(3);** //returnerer verdien '**r**'
  - **tekst1.equals(tekst2)** //returnerer **false**

Kopiering av tekster

- En String-variabel kan kopieres fra en annen
  - dette gjør ikke at objektet kopieres (samme som for array)
  - tekst2=tekst1; //peker til samme tekststreng

- Fordi tekstobjekter ikke kan endres (da lages det et nytt objekt), vil vi likevel ikke endre tekst1 om vi nå endrer tekst2
  - tekst2= tekst2.substring(0,5);

Sammenligning av tekster

- Uttrykket **(s==t)** er bare **true** dersom de peker på den *samme* tekststrengen.

- Skal vi teste på om to tekststrenger er *like* (men ikke nødvendigvis den samme) må vi bruke **(s.equals(t))** eller **(t.equals(s))**
  - (hva er forskjellen?)

Oppgave

- Skriv et program som leser inn en tekst i en String-variabel, og deretter plukker ut hvert 3. tegn i denne teksten. Disse lagres i en egen tekstvariabel, som skrives ut til slutt. Bruk String-metodene length() og charAt(), og konkatenering ved hjelp av +.
  - Hva skjer implisitt (uten at vi trenger instruere spesielt om det) ved bruk av + her?
  - Løs deretter oppgaven ved hjelp av substring() i stedet for charAt().

Løsning

```
String ekstraher (String s1) {
        String s2 = "";

        for (int i = 2; i<s1.length(); i=i+3)
            s2 = s2+s1.charAt(i);

        return s2;

  }

  String ekstraher2 (String s1) {
        String s2 = "";
        for (int i=2; i<s1.length(); i=i+3)
            s2 = s2+s1.substring (i,i+1);

        return s2;
  }
```

WHILE-LØKKER

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

- Utfører en samling programsetninger (blokk) så lenge en betingelse (logisk uttrykk) er oppfylt

```
while (fortsett inntil **) {
        ....
}
```

- Brukes for blokker som utføres til noe endrer seg, uansett hvor mange gjennomløp som kreves

EKSEMPLER!

FOR-LØKKER

- En annen måte å få utført en instruksjon (eller blokk) mange ganger er ved hjelp av en **for-løkke**
- Innebærer bruk av en variabel som teller (indeks)
- (begynn å telle på **; fortsett inntil **; øk telleren med ** for hver gjennomføring)
- Brukes for blokker som skal utføres et bestemt antall ganger (kan ligge i en variabel)
- Gir oss tilgang på en ”teller” (indeks)

Metoder – hvorfor?

- Ofte har vi bruk for å utføre (omtrent) samme instruksjoner flere steder i et program, eller i mange programmer. Gjenbruk
  - reduserer størrelsen på programmet
  - bedrer oversikten
  - forenkler vedlikeholdet
  - gir færre feil

- Da er det nyttig å kunne samle en eller flere instruksjoner med et selvvalgt navn, som så kan benyttes en eller flere ganger i programmet vårt

Metoder - eksempler

- Vi har brukt en del metoder allerede, for eksempel

```
System.out.println(" * ");
double d = tastatur.inDouble();
String s = tastatur.inWord();
int i = (int) Math.round(d);
```

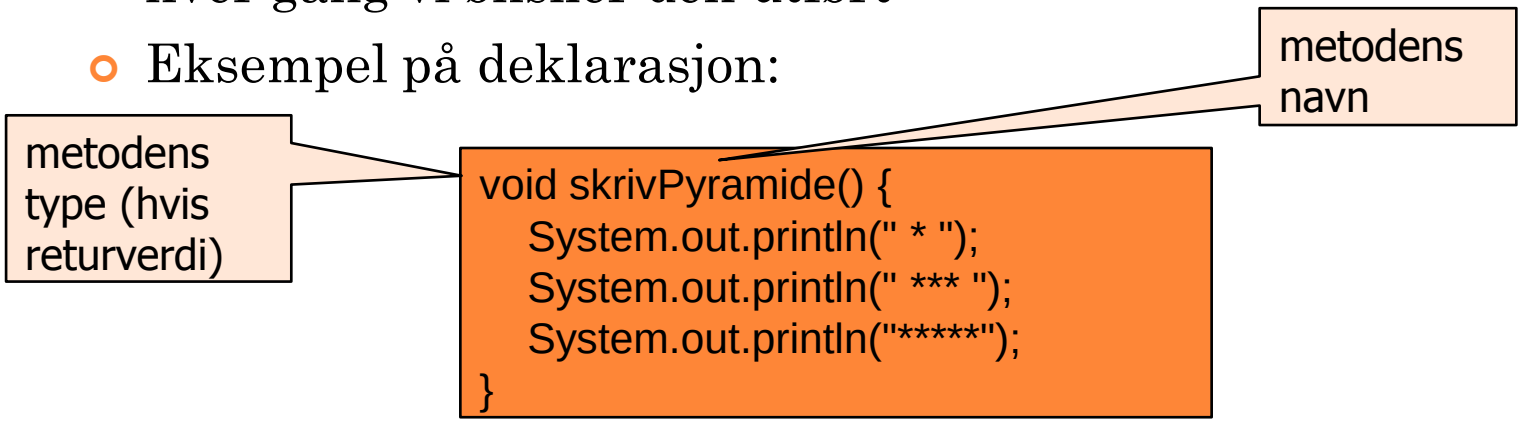
- Også main() er en metode:

```
public static void main(String[] args) {
…..
}
```

- Vi skal nå se nærmere på hva metoder egentlig er, hvordan de brukes og hvordan vi kan lage våre egne metoder.

METODER

- En metode er en navngitt blokk med instruksjoner (programsetninger) som vi får utført ved å angi metodens navn
- Den *deklareres* inne i en klasse, og kan senere *kalles på* hver gang vi ønsker den utført
- Eksempel på deklarasjon:



- Bruk av (kall på) metoden:

```
skrivPyramide();      // Her utføres setningen i metoden
```

Snartur innom konsepter

Objektorientert programmering

- Programmeringsspråk er designet etter ulike ”paradigmer” eller hovedprinsipper.
- Java er et objektorientert språk (andre typer er imperative eller funksjonelle)
- Det betyr at språket spesielt støtter programmereren i å tenke på det som skal beskrives og bearbeides i form av objekter – som har både egenskaper (variable) og handlinger (programsetninger samlet i metoder) knyttet til seg.
- Hvilke variable og metoder som hører til et objekt bestemmes av hvilken klasse objektet tilhører – en klasse beskriver et *mønster* for en type objekter.
- Viktige egenskaper ved objektorienterte programmer er muligheten til å skjule detaljer som ikke er vesentlig for omgivelsene (vi velger selv hva som skal synes og brukes fra utsiden av en klasse) og å gruppere variable og metoder som hører sammen.

Programstruktur med metoder

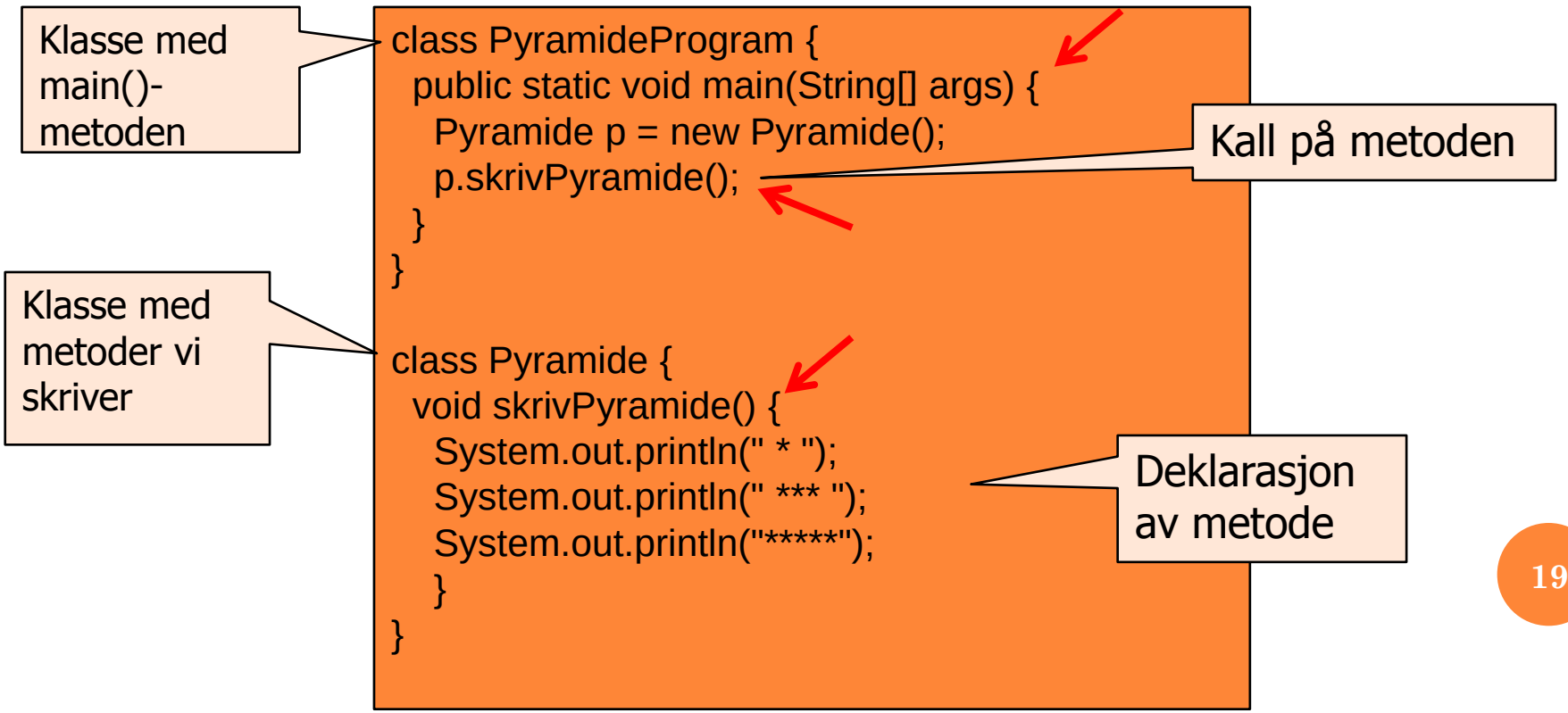
- Alle Java-programmer består av en eller flere klasser, med en eller flere metoder. Metoder deklareres inne i klasser.

- Hittil har programmene våre bestått av én klasse med metoden main() i. Klassen har samme navn som filen, og main() – som er der utføring av programmet starter og slutter - er den eneste metoden vi har skrevet, med all funksjonalitet i seg.

- Dere vil senere lære å skrive programmer med mange klasser og metoder – først skal vi imidlertid konsentrere oss om å lære deklarasjoner og bruk av metoder i en enkel, standard programstruktur
  - En klasse med main()-metoden, der program-utførelsen starter og avsluttes
  - En klasse med datastruktur, og metoder som bearbeider denne. Vi kan lage ett eller mange objekter fra denne klassen.

Enkel programstruktur med metoder

- Foreløpig vil vi benytte følgende mal for programmer med metoder (her et program fra filen PyramideProgram.java)



Oppgaver – Program 1

- Deklarer en klasse MetodeDemo med en metode demo. Metoden skal ikke ha parametere eller returnere noen verdi. Den skal utføre følgende:
  - deklarere to heltallsvariable og initiere dem med verdiene 3 og 14
  - deklarere en ny heltallsvariabel
  - gi den tredje variabelen verdien av summen av de to første
  - skrive ut verdien av den tredje variabelen
- Skriv et helt program med en egen klasse Program1 i tillegg til klassen MetodeDemo. Klassen Program1 skal inneholde metoden main som skal kalle på metoden demo i klassen MetodeDemo. (Du må da opprette et objekt av klassen MetodeDemo først).

Metode med returverdi: Eksempel

- Metoden **inInt()** leser inn et vilkårlig heltall. Noen ganger ønsker vi å sikre oss at vi får et **positivt** heltall. Vi kan da lage en egen metode for dette:

```
int lesPositivtHeltall() {

  In tastatur = new In();
  int tall;
  do {
    System.out.print ("Gi et positivt tall: ");
    tall = tastatur.inInt();
  } while (tall <= 0);

  return tall;
}
```

Deklarasjon av metoden

Eks: Program med bruk av metode med returverdi

```
import easyIO.*;
class TestPosTall {
    public static void main(String[] args) {
            PosTall pt = new PosTall();

            int i = pt.lesPositivtHeltall();
            System.out.println(i + " er et positivt heltall");
    }
}
class PosTall {
    int lesPositivtHeltall() {
            In tastatur = new In();
            int tall;
            do {
                System.out.print("Gi et positivt tall: ");
                tall = tastatur.inInt();
            } while (tall <= 0);
            return tall;
    }
}
```

Metoder med parametere

- Ofte ønsker vi at samme metode skal kunne brukes for litt ulike input-verdier, f. eks:

```
System.out.println(" * ");

System.out.println("Hei verden");
```

- Her er println() en metode som tar en tekst som input (**parameter)**

- Metoden gjør det samme uansett hvilken String-verdi vi kaller med: Skriver den ut på skjermen

Eks: Metode med parameter

- Parameteren avgjør antall linjer i ”pyramiden”:

```
class PyramideProgram {
    public static void main(String[] args) {
            Pyramide p = new Pyramide();
            p.skrivPyramide(4);
    }
}
class Pyramide {
    void skrivPyramide(int antall) {

            for (int i = 1; i<=antall; i++) {
                for (int j = 1; j<=i; j++) {
                        System.out.print("*");
                }
                System.out.println ();
            }
    }
}
```

parameter angis med type og navn som skal benyttes inni metoden

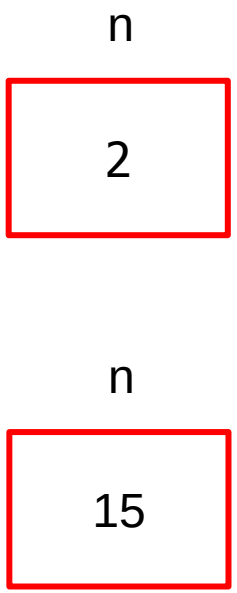
Eks: Gangetabell-metode

- Metode som skriver ut n-gangen fra 1 til 10
- deklarasjon

```
void gangeTabell(int n) {
    for (int i = 1; i <= 10; i++) {
        System.out.println(i * n);
    }
  }
```

- Eksempler på kall på metoden

```
gangeTabell(2);
gangeTabell(15);
```



Eks: Bruk av gangetabell-metode

```
import easyIO.*;
class GangeProgram {
    public static void main(String[] args) {
            In tastatur = new In();
            Utregning utr = new Utregning();
            System.out.print("Hvilken gangetabell vil du skrive? ");
            int tall = tastatur.inInt();
            utr.gangeTabell(tall);
    }
}
class Utregning {
    void gangeTabell(int n) {
            for (int i = 1; i <= 10; i++) {
                System.out.println(i * n);
            }
    }
}
```

Oppgave

- Skriv et program som deklarerer og bruker denne metoden.
- Initialiser arrayen selv før du kaller metoden – skriv ut summen metoden returnerer med.

```
double finnSum (double[] x) {
   double sum = 0.0;
   for (int i=0; i<x.length; i++) {
      sum += x[i];                  // sum=sum+x[i];
   }
   return sum;
}
```

Oppgave - løsningsforslag

```
class TestMetode {
    public static void main (String [] args) {
                double [] a = {1.3, 2.0, 7.5, 10};

                MetodeKlasse mk = new MetodeKlasse ();
                double total = mk.finnSum(a);
                System.out.println ("Returverdi = " + total);
    }
}

class MetodeKlasse {
    double finnSum(double[] x) {
                double sum = 0.0;
                for (int i=0; i<x.length; i++) {
                    sum += x[i];
                }
                return sum;
    }
}
```

Oppsummering metoder: Deklarasjon

- Java-programmene så langt i kurset består av to klasser, startklassen med main og en annen klasse hvor det kan det befinne seg en eller flere metoder.
- De metodene vi ser på så langt i kurset har følgende form:

type verdi metoden returnerer, f.eks. int, double, char, ... eller void hvis ingen verdi

et navn vi velger

Beskrivelse av hva slags input metoden skal ha - i form av variabel-deklarasjoner skilt av komma

```
returverditype metodenavn (parametre) {
        setninger 1;
        setninger 2;
        ....
        setninger n;
        return verdi;    // Hvis ikke returverditype void
}
```

Oppgaver II – Program 1

- Skriv om metoden demo til å ta to heltalls-paramere og erstatt bruken av de to første variablene med disse
- Skriv om metoden demo til å returnere verdien av den tredje variabelen
- Endre kallet på metoden demo i main-metoden så det passer til endringene (to parametere og en returverdi). Du skal kalle metoden med argumentene 3 og 13, og skrive ut hvilken verdi metoden returnerer før programmet avsluttes.