Uke 4, INF 1000, 13 sept. 2005 - Løkker og arrayer.

Institutt for Informatikk
Universitet i Oslo

Arild Waaler

Blokker

- En blokk er en samling instruksjoner omgitt av krøllparenteser:

```
{
 instruksjon 1;
 instruksjon 2;
 ....
 instruksjon n;
}
```

- Alle steder i et Java-program hvor det kan stå en instruksjon, kan vi om ønskelig i stedet sette en blokk.

Eksempel

```
class Blokker {
   public static void main (String [] args) {
      double x = -10.2;
      double absx;

      if (x < 0) {
         absx = -x;
         System.out.println("Absoluttverdien er: " + absx);
      } else {
         absx = x;
         System.out.println("Absoluttverdien er: " + absx);
      }
   }
}
```

her er en blokk

her er en blokk

Deklarasjoner inne i blokker

- Vi har lov til å deklarere variable inne i en blokk, forutsatt at de ikke allerede er deklarert utenfor blokken. Eksempel:

```
double x = 0.3;
if (x < 0) {
 double y;   // y er deklarert inne i en blokk
 y = -x;
}
```

- En slik variabel (f.eks. y ovenfor) slutter å eksistere når blokken er utført. Derfor er dette ulovlig:

```
double x = 0.3;
if (x < 0) {
 double y;
 y = -x;
}
x = y; // Ulovlig, siden y ikke eksisterer her
```

while-setninger

- Vi kan få utført en instruksjon (eller en blokk) mange ganger ved hjelp av en while-setning (også kalt while-løkke):

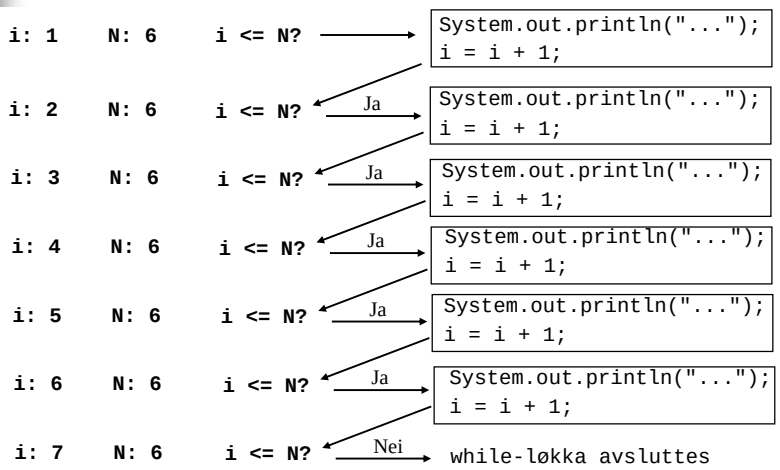
```
while (logisk uttrykk) {
       setning 1;
       setning 2;
       .....
       setning n;
}
```

- Virkemåte:
  1. Det logiske uttrykket regnes ut
  2. Hvis uttrykket er sant (true), utføres setning 1,2,...,n, og deretter går vi til punkt 1 ovenfor (NB: det logiske uttrykket kan da ha skiftet verdi!)
  3. Hvis uttrykket er usant (false), avsluttes while-setningen.

Eksempel på bruk av while-setning

```
class SkrivLinjer {
   public static void main (String [] args) {
        int N = 6;
        int i = 1;

        while (i <= N) {
               System.out.println("Jeg må lese til eksamen");
               i = i + 1;
        }

        System.out.println("Nå er while-løkka ferdig");

   }
}
```


Hva skjer når while-setningen utføres?



Kompilering og kjøring

```
> javac SkrivLinjer.java
> java SkrivLinjer
Jeg må lese til eksamen
Jeg må lese til eksamen
Jeg må lese til eksamen
Jeg må lese til eksamen
Jeg må lese til eksamen
Jeg må lese til eksamen
Nå er while-løkka ferdig
```

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 EvigLokke1 {
  public static void main (String [] args) {
    while (true) {
      System.out.println("INF 1000");
    }
  }
}
```



```
class EvigLokke2 {
  public static void main (String [] args) {
    int i = 1, j = 2;
    while (i < j) {
      System.out.println("Nå er i < j");
    }
  }
}
```


Kompilering og kjøring

```
> javac EvigLokke1.java
> java EvigLokke1
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
   ...
   ...
   ...
 (osv)
```

```
> javac EvigLokke2.java
> java EvigLokke2
Nå er i < j
Nå er i < j
Nå er i < j
Nå er i < j
Nå er i < j
Nå er i < j
   ...
   ...
   ...
 (osv)
```


Oppgave

- Hva blir utskriften fra dette programmet?


```
class LokkeTest {
  public static void main (String [] args) {
    int i = 3;

    while (i > 0) {
      System.out.print("Nå er i = ");
      System.out.println(i);
      i = i - 1;
    }
  }
}
```


Oppgave 2

- Hva blir utskriften fra dette programmet?


```
class LokkeTest2 {
  public static void main (String [] args) {
    int i = 1;

    while (i < 4) {
      System.out.print("Nå er i = ");
      System.out.println(i);
      i = i + 1;
    }
  }
}
```

Eksempel: gangetabell

- Problem:

  Lag et program som bruker en while-løkke til å beregne "tregangen" og lage en slik utskrift på skjermen:

  1 * 3 = 3
  2 * 3 = 6
  3 * 3 = 9
  4 * 3 = 12
  5 * 3 = 15
  6 * 3 = 18
  7 * 3 = 21
  8 * 3 = 24
  9 * 3 = 27
  10 * 3 = 30

Løsningsskisse

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

    <deklarasjoner>

    <initialisering av variable>

    <while-løkke, hvor hvert gjennomløp
     regner ut og skriver ut en ny linje
     på skjermen>

  }
}
```


while-løkka

```
while (ikke ferdig) {
  <finn svaret på neste
  regnestykke>
  <skriv ut svaret>
}
```

- For å finne svaret på neste regnestykke, må vi holde rede på de to tallene som skal ganges sammen. Vi trenger bare en variabel for dette, siden annen faktor alltid er lik 3. Vi lager også en variabel for svaret:

  int k, svar;

- Vi initialiserer slik:

  k = 1;

- Dermed blir svaret på neste regnestykke:

  svar = k * 3;

while-løkka *forts.*

```
while (ikke ferdig) {
   int svar = k * 3;
   System.out.print(k + " * 3 = " + svar);
}
```

- Vi må i tillegg huske å endre verdien til k i slutten av hvert gjennomløp (ellers gjør vi samme regnestykke igjen og igjen):

  k = k + 1;

- Hvor lenge skal while-løkka løpe ? Vi kan bruke denne testen:

  k <= 10

Ferdig program

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

      int k=1;
      while (k <= 10) {
         int svar = k * 3;
         System.out.println(k + " * 3 = " + svar);
         k = k + 1;
      }

   }
}
```


Kompilering og test

```
> javac TreGangen.java
> java TreGangen
1 * 3 = 3
2 * 3 = 6
3 * 3 = 9
4 * 3 = 12
5 * 3 = 15
6 * 3 = 18
7 * 3 = 21
8 * 3 = 24
9 * 3 = 27
10 * 3 = 30
```


Eksempel: stolpediagram

- Problem:

  Stolpediagrammer brukes i mange sammenhenger for å visualisere forskjeller i størrelse, vekt, lengde eller liknende. Anta f.eks. at fire mobiltelefoner koster henholdsvis 600, 300, 900 og 1200 kroner. Det kan vi visualisere slik:

  mobil 1: ******
  mobil 2: ***
  mobil 3: *********
  mobil 4: ************

  Antall stjerner er i hvert tilfelle lik prisen / 100.

  Vi skal lage et program som lager et slikt stolpediagram for to mobiltelefoner, gitt prisen på de to telefonene.

Hvilke data beskriver problemet?

- Input:
  - Prisen til de to telefonene (to desimaltall)
- Output:
  - Antall stjerner som skal skrives ut for hver av telefonene (to heltall)

Fremgangsmåte

- Antall stjerner som skal skrives ut for hver telefon er prisen delt på 100. Vi må også huske å avrunde til et heltall.

  Dette kan vi f.eks. gjøre slik:

  ```
  antall1 = (int) (pris1 / 100);
  antall2 = (int) (pris2 / 100);
  ```

- For å få skrevet ut riktig antall stjerner bruker vi en while-løkke:
  - Hvert gjennomløp av while-løkka skriver ut en enkelt stjerne.
  - Vi teller opp antall gjennomløp av løkka (= antall stjerner) og stopper når antallet er det vi ønsker.

Løsningsskisse

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

    <les inn pris1 og pris2 fra terminal>

    antall1 = (int) (pris1 / 100);
    <skriv ut "mobil 1: ">
    <skriv ut antall1 stjerner>

    antall2 = (int) (pris2 / 100);
    <skriv ut "mobil 2: ">
    <skriv ut antall2 stjerner>

  }
}
```


Hvordan skrive ut antall1 stjerner?

- Vi kan bruke en while-løkke:

```
int i = 0;
while (i < antall1) {
   System.out.print("*");
   i = i + 1;
}
```

- En ting mangler ovenfor: vi må huske å lage et linjeskift etter at alle stjernene er skrevet ut (slik at neste stolpediagram kommer på ny linje). Vi kan gjøre dette slik (etter while-løkka):

```
System.out.println("");          // Denne lager linjeskift
```



```
import easyIO.*;

class StolpeDiagram {
  public static void main (String[] args) {
    In tast = new In();

    System.out.print("Pris på telefon 1: ");
    double pris1 = tast.inDouble();
    System.out.print("Pris på telefon 2: ");
    double pris2 = tast.inDouble();

    System.out.print("mobil 1: ");
    int antall1 = (int) (pris1 / 100);
    int i = 0;
    while (i < antall1) {
      System.out.print("*");
      i = i + 1;
    }
    System.out.println();     // Start ny linje på skjermen

    System.out.print("mobil 2: ");
    int antall2 = (int) (pris2 / 100);
    i = 0;
    while (i < antall2) {
      System.out.print("*");
      i = i + 1;
    }
    System.out.println();
  }
}
```

Kompilering og test

```
> javac Stolpediagram.java
> java Stolpediagram
Pris på telefon 1: 800
Pris på telefon 2: 1200
mobil 1: ********
mobil 2: **********
```

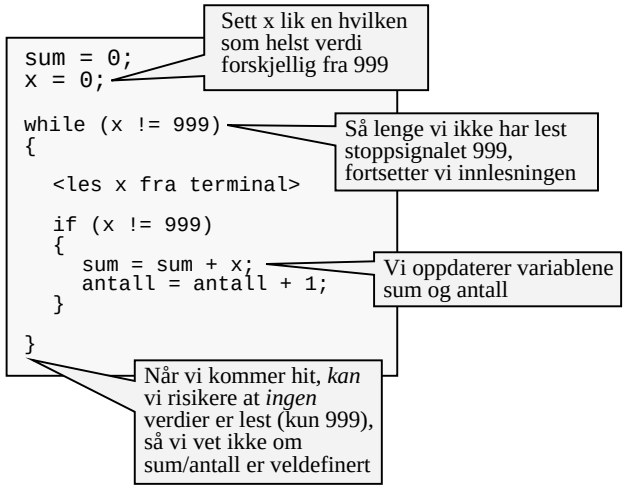

Eksempel: finne gjennomsnitt

- Problem:

  Lag et program som leser en rekke desimaltall fra terminal, helt til brukeren oppgir tallet 999 som betyr at innlesningen skal avsluttes. Programmet skal deretter regne ut gjennomsnittet av de innleste verdiene og skrive ut svaret på skjermen.

- Framgangsmåte:
  - Vi bruker inDouble() til å lese desimaltallene
  - Vi lager en while-løkke for innlesningen, slik at innlesningen kan foretas så mange ganger det er ønsket
  - Hver gang vi leser en ny lovlig verdi, adderer vi det til summen av de foregående verdiene. Vi holder også rede på hvor mange verdier som er innlest.
  - Testen i while-løkka sørger for stopp når siste innleste verdi er 999.
  - Etter while-løkka deler vi summen på antall innleste verdier, og skriver ut svaret på skjermen.

While-løkka



Ferdig program

```
import easyIO.*;

class FinnGjennomsnitt {
  public static void main (String[] args) {
    double x = 0, sum = 0;
    int antall = 0;
    In tast = new In();
    Out skjerm = new Out();

    while (x != 999) {
      skjerm.out("Oppgi et desimaltall (999 for å avslutte): ");
      x = tast.inDouble();
      if (x != 999) {
        sum = sum + x;
        antall = antall + 1;
      }
    }

    if (antall == 0) {
      skjerm.outln("Ingen verdier ble oppgitt!");
    } else {
      skjerm.out("Gjennomsnitt: ");
      skjerm.outln(sum/antall, 2);
    }

  }
}
```

Eksempel: innlesning med sjekk

- Problem:

  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.

- Framgangsmåte:
  - Vi bruker metoden inInt() til å lese heltallet fra terminal
  - Vi legger selve innlesningen inni en while-løkke, slik at innlesningen om nødvendig kan utføres flere ganger
  - Testen øverst i while-løkka må være true når vi første gang kommer til while-løkka, og vi må sørge for at verdien blir false straks vi har lest en lovlig verdi (slik at videre innlesning stopper)

Programskisse

```
import easyIO.*;

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

    <deklarasjoner>

    boolean fortsett = true;

    while (fortsett)
    {
      <les inn heltall fra terminal>

      <sett fortsett lik false hvis lovlig verdi,
       og gi feilmelding hvis ulovlig verdi>
    }

    <skriv ut verdien>
  }
}
```


Ferdig program

```
import easyIO.*;

class LesVerdi {
  public static void main (String[] args) {
    int verdi = 0;
    boolean fortsett = true;
    In tast = new In();

    while (fortsett)
    {
      System.out.print("Oppgi verdi (1,2,...,100): ");
      verdi = tast.inInt();

      if (verdi >= 1 & verdi <= 100)
      {
        fortsett = false;
      }
      else
      {
        System.out.println("Ulovlig verdi! Prøv igjen!");
      }
    }

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


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-setninger

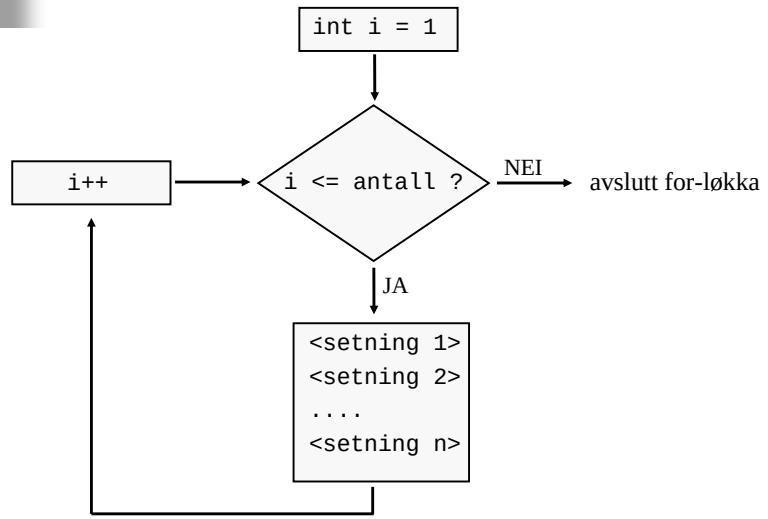
- En annen måte å få utført en instruksjon (eller blokk) mange ganger er ved hjelp av en for-setning (også kalt for-løkke):

initialisering
løkketest
løkkeoppdatering

```
for (int i=1; i<=antall; i++)
{
   <setning 1>
   <setning 2>
   ....
   <setning n>
}
```

1. Først deklareres tellevariabelen i og settes lik 1.
2. Hvis i <= antall gå til punkt 3; hvis ikke avsluttes for-setningen.
3. Utfør setning 1, 2, ..., n
4. Utfør i++ og gå til punkt 2.

Hvordan for-setningen virker



Operatorene ++ og --

Instruksjon	Alternativ 1 Prefiks-operator	Alternativ 2 Postfiks-operator
`i = i + 1`	`++i`	`i++`
`i = i - 1`	`--i`	`i--`

- `++i, i++, --i` og `i--` endrer ikke bare på verdien til i, de er dessuten uttrykk som selv har en verdi. Dermed kan vi f.eks. skrive:

```
System.out.println(i++);  // Skriv ut i og øk deretter i med 1
System.out.println(++i);  // Øk i med 1 og skriv deretter ut i
```

- Prefiks-operatorene endrer verdien til variabelen før uttrykket er evaluert.
- Postfiks-operatorene endrer verdien etter at uttrykket er evaluert.

Eksempel

Programkode	Verdien til k	Verdien til m	Verdien til n

`int k = 0;`
`int m;`
`int n;`
	0	udefinert	udefinert

`k = k + 1;`
`m = ++k;`
`n = k++;`
	3	2	2

```
k = k + 1;
{ k = k + 1;
  m = k;
{ n = k;
  k = k + 1;
```

Nøtt

- Hva blir utskriften fra dette programmet?

```
class InkrementOperator {
   public static void main (String [] args) {
       for (int i=0; i<10; i++) {
         int j = i;
         int k = j++ + ++j;
         System.out.println("k = " + k);
       }
   }
}
```


Hva skjedde?

- Første løkkegjennomløp (i = 0):
  - **`int j = i;`**
    Variabelen j settes lik 0.
  - **`int k = j++  + ++j;`**
    j++ gir verdien **0** og øker j til 1
    ++j øker j til 2 og gir verdien **2**
    k settes lik 0 + 2, altså 2.
- Andre løkkegjennomløp (i = 1):
  - **`int j = i;`**
    Variabelen j settes lik 1.
  - **`int k = j++ + ++j;`**
    j++ gir verdien **1** og øker j til 2
    ++j øker j til 3 og gir verdien **3**
    k settes lik 1 + 3, altså 4.
- Osv.

Nesting av løkker

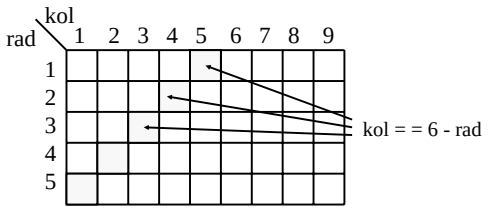
- Det er ofte behov for å neste løkke-setninger inne i hverandre; vi kommer til å se mange eksempler etterhvert.
- Eksempel på nestet for-løkke:

```
for (int i=0; i<10; i++) {
       for (int j=0; j<10; j++) {
       int produkt = i * j;
       System.out.println("i * j = " + produkt);
       }
}
```


Nesting av løkker: større eksempel

- Vi kan bruke nestede løkker til å produsere følgende utskrift på skjermen:

```
1
212
32123
4321234
543212345
```



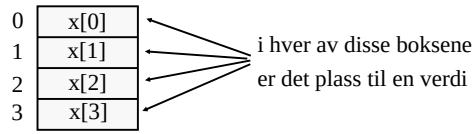
- Ytre løkke kontrollerer linjene; dvs hvert gjennomløp av den ytre løkka skal produsere en ny linje med utskrift.
- Indre løkke kontrollerer de enkelte tegnene på en linje; dvs hvert gjennomløp av den indre løkka skal skrive et nytt siffer.

Programmet

```
class SkrivPyramide {
  public static void main (String [] args) {
    for (int rad = 1; rad < 6; rad++) {

      for (int kol = 1; kol < 6 - rad; kol++) {
        System.out.print(" ");
      }

      for (int sif = rad; sif >= 1; sif--) {
        System.out.print(sif);
      }

      for (int sif = 2; sif <= rad; sif++) {
        System.out.print(sif);
      }

      System.out.println();
    }
  }
}
```


Arrayer

- Hittil har vi sett på variable som kan holde en enkelt verdi:
  - en int-variabel har plass til ett heltall
  - en String-variabel har plass til en enkelt tekststreng
  - osv.
- Arrayer er "variable" som kan holde på mange verdier:
  - en int-array har plass til mange heltall
  - en String-array har plass til mange tekststrenger
  - osv.
- Verdiene som ligger i en array har hver sin posisjon (= indeks):
  0, 1, 2, ..., K-1 hvor K = lengden til arrayen
- En array x med lengde 4 kan visualiseres slik:



Deklarere og opprette arrayer

- Deklarere en array (gi den et navn):

```
datatype[] arrayNavn;
```

- Opprette en array (sette av plass i hukommelsen):

```
arrayNavn = new datatype[K];   // K er ønsket lengde
```

- Deklarere og opprette i en operasjon:

```
datatype[] arrayNavn = new datatype[K];
```

- Eksempler:

```
int[] a = new int[10];
double[] x = new double[100];
String[] s = new String[1000];
```


Verdiene i en array

- Anta at vi har deklarert og opprettet følgende array:

```
int[] tlf = new int[600];
```

- For å få tak i de enkelte verdiene i arrayen:

```
tlf[0], tlf[1], tlf[2], ..., tlf[599]
```

- For å få tak i lengden på arrayen:

```
tlf.length      // NB: ingen parenteser til slutt
```

- For å sortere elementene i en array (i stigende rekkefølge):

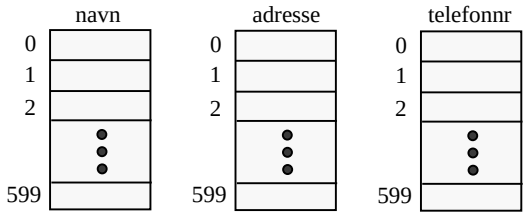
```
java.util.Arrays.sort(tlf);
```

Eksempel på bruk av arrayer

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

```
String[] navn = new String[600];
String[] adresse = new String[600];
int[] telefonnr = new int[600];
```

- Resultatet kan visualiseres slik:



Automatisk initialisering av arrayer

- Når en array blir opprettet, blir den automatisk initialisert (dvs verdiene er ikke udefinerte når den er opprettet).
  - `int[] k = new int[100];` // Nå er alle k[i] == 0
  - `double[] x = new double[100];` // Nå er alle x[i] == 0.0
  - `boolean[] b = new boolean[100];` // Nå er alle b[i] == false
  - `char[] c = new char[100];` // Nå er alle c[i] == '\u0000'
  - `String[] s = new String[100];` // Nå er alle s[i] == null
- Merk: String-arrayer initialiseres med den spesielle verdien **`null`**. Dette er *ikke* en tekststreng og må ikke blandes sammen med en tom tekst: **`""`**.
- For å kunne bruke verdien **`s[i]`** til noe fornuftig må du først sørge for å gi **`s[i]`** en tekststreng-verdi, f.eks. **`s[i] = "Per"`** eller **`s[i] = "".`**

Egendefinert initialisering av en array

- Det er ikke alltid den automatiske initialiseringen av en array gir det vi ønsker. Vi kan da initialisere arrayen med våre egne verdier, slik som i disse eksemplene:

```
int[] primtall = {2, 3, 5, 7, 11, 13};

double[] halve = {0.0, 0.5, 1.0, 1.5, 2.0};

String[] ukedager = {"Mandag", "Tirsdag",
      "Onsdag", "Torsdag", "Fredag", "Lørdag",
      "Søndag"};
```


Eksempel: lese og skrive ut

- Program som leser fem navn fra terminal og skriver dem ut igjen:


```
import easyIO.*;

class LesOgSkriv {
  public static void main (String [] args) {
     In tastatur = new In();
     String[] s = new String[5];
     for (int i=0; i<5; i++) {
        System.out.print("Navn: ");
        s[i] = tastatur.inLine();
     }

     for (int i=0; i<5; i++) {
        System.out.println(s[i]);
     }
  }
}
```

Eksempel: lese og skrive ut sortert

- Program som leser fem navn fra terminal og skriver dem ut i sortert rekkefølge:

```
import easyIO.*;

class LesOgSorter {
  public static void main (String [] args) {
     In tastatur = new In();
     String[] s = new String[5];
     for (int i=0; i<5; i++) {
        System.out.print("Navn: ");
        s[i] = tastatur.inLine();
     }
     java.util.Arrays.sort(s);
     for (int i=0; i<5; i++) {
        System.out.println(s[i]);
     }
  }
}
```

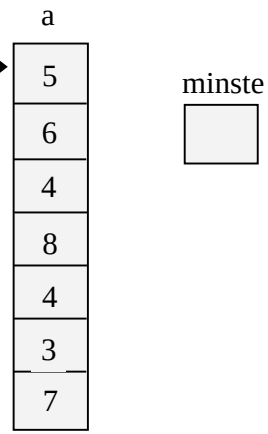

Eksempel: finne en bestemt verdi

```
/* a er en int-array og x er en int-variabel.  Vi ønsker å sjekke om verdien i x
   forekommer i a */
boolean funnet = false;
int i = 0;
while (i<a.length && !funnet) {
  if (a[i] == x) {
    funnet = true;
  }
  i++;
}
if (funnet) {
  System.out.println("Verdien ligger i posisjon " + (i-1));
} else {
  System.out.println("Verdien ble ikke funnet!");
}
```

Eksempel: finne den minste verdien

- Følgende eksempel illustrerer hvordan man kan finne den minste verdien i en array:

```
//Anta at x er en double-array
double minVerdi = a[0];
int minPos = 0;
for (int i=1; i<a.length; i++) {
  if (a[i] < minVerdi) {
    minVerdi = a[i];
    minPos = i;
  }
}
// Nå er minste lik den minste verdien i x
// og minPos er posisjonen den ligger i
```



Eksempel: ferdig program

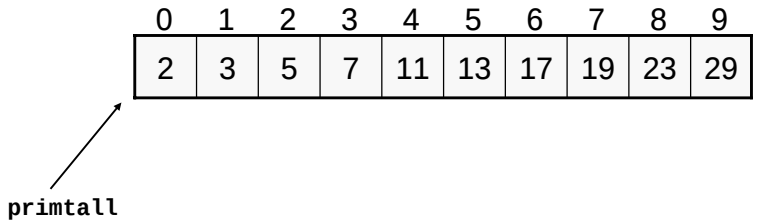
```
import easyIO.*;

class FinnMinsteVerdi {
   public static void main (String [] args) {
      In tastatur = new In();
      double[] a = new double[5];

      for (int i=0; i<a.length; i++) {
         System.out.print("Oppgi en verdi: ");
         a[i] = tastatur.inDouble();
      }

      double minVerdi = a[0];
      int minPos = 0;
      for (int i=1; i<a.length; i++) {
         if (a[i] < minVerdi) {
            minVerdi = a[i];
            minPos = i;
         }
      }
      System.out.println("Minste verdi er " + minVerdi);
      System.out.println("Den ligger på plass " + minPos);
   }
}
```

En array-variabel er en adresse

- Når vi deklarerer en array så refererer arraynavnet ikke til selve verdiene i arrayen, men til adressen (i hukommelsen) hvor verdiene ligger lagret.
- Resultatet etter at vi har utført

```
int[] primtall = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
```

kan visualiseres slik:



Å kopiere en array-adresse

- Hvis vi har

```
double [ ]  x = new double [100];
```

så vil

```
double [ ]   y = x;
```

medføre at adressen til arrayen vi opprettet kopieres over til variabelen y (dermed har vi fortsatt bare ett sett med verdier lagret, men vi har to referanser til arrayen: x og y.

Oppgave

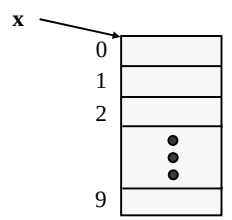
Hva blir utskriften fra følgende program?

```
class ToArrayer {
 public static void main (String [] args) {
   int[] x = new int[10];
   int[] y = x;

   for (int i=0; i<10; i++) {
    x[i] = i;
   }

   for (int i=0; i<10; i++) {
    System.out.println(y[i]);
   }
 }
}
```

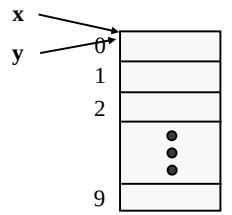

Hva som skjedde

Etter å ha utført instruksjonen ..... **.... så er situasjonen denne:**

```
int[] x = new int[10];
```




```
int[] y = x;
```



Kompilering og kjøring

```
> javac ToArrayer.java
> java ToArrayer
0
1
2
3
4
5
6
7
8
9
```


Kopiering av arrayer

Vi kan ikke lage en kopi av en array x ved å skrive

```
int[] y = x;
```

siden dette bare medfører at adressen til arrayen legges inn i y.

- Skal vi lage en kopi, må vi først opprette en array til (f.eks. y), og så kopiere over verdiene en for en:

```
double[] y = new double[x.length];
for (int i=0; i<x.length; i++) {
    y[i] = x[i];
}
```

- Det finnes også ferdige verktøy i Java for å kopiere en array, f.eks:

```
int[] y = (int[]) x.clone();
```


Når arrayen blir for liten

- Noen ganger får vi behov for å utvide en array.
- Framgangsmåte for å utvide en array:
  - Deklarer og opprett en ny array **`temp`** som er av ønsket lengde
  - Flytt over alle verdier fra opprinnelig array til **`temp`**
  - Sett opprinnelig array-peker til å peke på **`temp`**
- Programkode:

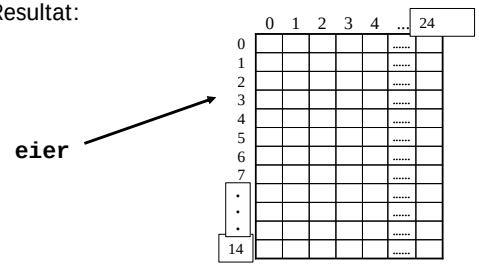
```
/* Anta at tall er en int-array og at vi ønsker
   å utvide tall til dobbel lengde */

int[] temp = new int[2 * tall.length];
for (int i=0; i<tall.length; i++) {
   temp[i] = tall[i];
}
tall = temp;
```


Flerdimensjonale arrayer

- Vi kan også deklarere todimensjonale (og høyeredimensjonale) arrayer.
- Eksempel:

```
String[][] eier = new String[15][25];
```

- Resultat:



Hvis et felt ikke er solgt:

```
eier[i][j] = = null
```

Hvis et felt er solgt:

```
eier[i][j] ! = null
```

- Eksempler på lovlige operasjoner:

```
eier[3][4] = "Petrol A/S";
int antallRader = eier.length;
int antallKolonner = eier[0].length;
```

Eksempel: finn antall solgte oljefelt

```
class AntallFelt {
 public static void main (String [] args) {
   String[][] eier = new String[15][25];
   <innlesning av eiere m.m.>

   int antallSolgte = 0;
   for (int i=0; i<15; i++) {
     for (int j=0; j<25; j++) {
       if (eier[i][j] != null) {
         antallSolgte++;
       }
     }
   }
   System.out.println(”Antall solgte felt: ” + antallSolgte);
 }
}
```