UNIVERSITETET I OSLO

INF1000 høst 2010

Forelesning 2:
- Innlesning fra terminal
- Boolean-variable
- if-setninger
- Løkker
- Litt mer om heltall: divisjon og modulo
- Arrayer

UNIVERSITETET I OSLO

Innlesning fra tastatur med easyIO

- Vi må først skrive i toppen av programmet:
```
import easyIO.*;
```
- Inne i klassen deklarerer vi en variabel av type In:
```
In tastatur = new In();
```
- Så kan vi bruke variabelen vi har deklarert til å lese inn fra tastatur, f.eks. et heltall:
```
int radius;
System.out.print("Oppgi radiusen: ");
radius = tastatur.inInt();
```



```
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);
  }
}
```

Vi importerer pakken easyIO

Vi oppretter en "verktøy-kasse" (In) for lesing fra terminal og deklarerer en variabel som vårt håndtak til verktøykassen.

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

UNIVERSITETET I OSLO

```
$ javac   LesFraTerminal.java
$ java   LesFraTerminal
Skriv et heltall: 123
Du skrev: 123

$
```

Lesemetoder i easyIO

```
// 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();
```

char er typen til en bokstav. Mer neste uke.

Eksempel på innlesning

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

- Lag et program som leser fra terminal disse dataene om en person:
  - Navn
  - Yrke
  - Alder

  og som skriver ut dataene på skjermen etterpå
- Framgangsmåte:
  - **inLine()** for å lese navn og yrke
  - **inInt()** for å lese alder

```
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);
  }
}
```

```
$ javac  LesDataOmPerson .java
$ java  LesDataOmPerson
Navn: Oluf
Yrke: Komiker
Alder: 66
Hei Oluf, du er 66 år gammel og jobber som Komiker
$
```

Datatypen boolean

- En boolean-variabel kan kun ha verdiene true eller false:

```
boolean akseptabeltTilbud;
```

- Den kan gis verdi ved et logisk uttrykk

```
akseptabeltTilbud = (tilbud < 1000);
```

- akseptabeltTilbud vil nå være sann eller usann avhengig av verdien til tilbud

Boolean-uttrykk

Uttrykket	har verdien true hvis
`x < y`	`x mindre enn y`
`x <= y`	`x mindre enn eller lik y`
`x == y`	`x lik y`
`x != y`	`x ikke lik y`
`x > y`	`x større enn y`
`x >= y`	`x større enn eller lik y`
`!(x < y)`	`ikke x mindre enn y`
`b1 && b2`	`både b1 og b2 sann`
`b1 || b2`	`b1 eller b2 (eller begge) sann`

Greit å vite om boolean

- Det er forskjell på = og = = :
  - `=` brukes for å sette verdien til en variabel
  - `= =` brukes for å sammenlikne to verdier
- Hvis vi har variabelen `boolean b` så er det ingen forskjell på

```
b == true
b
```

- Ekstra parenteser kan øke leseligheten:

```
b = x == y;     betyr det samme som
b = (x == y);
```

Blokker

- En **programblokk** er en samling med programsetninger omsluttet av krøllparenteser
- Setningene i main-metoden ligger inne i en blokk
- Blokker kan **nøstes** inne i hverandre
  - vi kan ha blokker inne i blokker
- En variabel deklarert inne i en blokk er kun definert ("synlig") fra stedet den er deklarert til slutten av blokken
- Vi kaller dette **skopet** til variabelen

Eksempel som kompilerer

```
class SkopLovlig {
    public static void main(String[] args){
        int k = 15;
        {
            int n = 10;
            System.out.println(k + n);
        }
        // Her er ikke n definert
       // System.out.println(n) vil gi feil
        System.out.println(k);
   }
}
```

Eksempel som ikke kompilerer

```
class SkopIkkeLovlig {
  public static void main(String[] args){
    int k = 15;
    {
      int n = 10;
      int k = 200; // Ikke lov
      // k er allerede definert
    }
  }
}
```

```
$ javac SkopIkkeLovlig.java
SkopIkkeLovlig.java:6: k is already defined in
  main(java.lang.String[])
            int k = 200; // Ikke lov.
                ^
1 error
$
```

if-setninger (forgreninger)

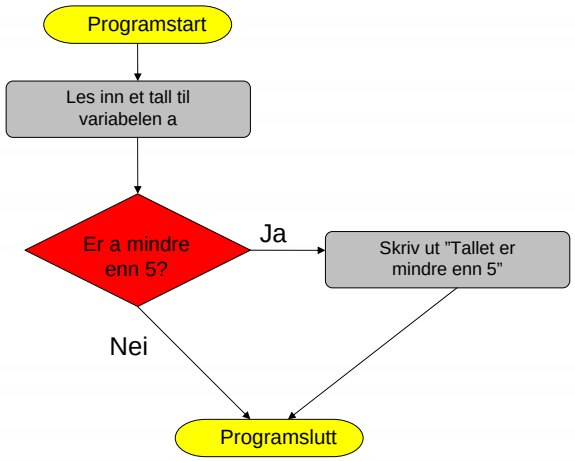
```
if (logisk uttrykk)
{
     <setninger>
}
else
{
     <setninger>
}
```

Et boolean-uttrykk, f.eks. x < y

Den første blokken (og bare den) blir utført hvis det logiske uttrykket er sant (true)

Den andre blokken (og bare den) blir utført hvis det logiske uttrykket er usant (false)

Flytdiagram for eksempel



Flytdiagram implementert

```
import easyIO.*;
class LesTall {
    public static void main(String args[]){
        In tastatur = new In();
        int a = tastatur.inInt();
        if(a<5){
            System.out.println(
                  "Tallet er mindre enn 5");
        }
    }
}
```

```
$ javac LesTall.java
$ java LesTall
4
Tallet er mindre enn 5
$
```

```
import easyIO.*;
class Hoyde {
    public static void main (String[] args) {
        In tastatur = new In();
        double hoyde1, hoyde2;
        System.out.print("Høyden til Per: ");
        hoyde1 = tastatur.inDouble();
        System.out.print("Høyden til Kari: ");
        hoyde2 = tastatur.inDouble();

        if (hoyde1 > hoyde2) {
            System.out.println("Per er høyere enn Kari");
        } else {
            System.out.println("Per er ikke høyere enn Kari");
        }
    }
}
```

```
$ javac   Hoyde.java
$ java   Hoyde
Høyden til Per: 178
Høyden til Kari: 178
Per er ikke høyere enn Kari
$
```

Varianter av if-setninger

- else-delen kan utelates:

```
if (pris > 1500){
     System.out.println(”For dyrt”);
}
```

- Klammene også (hvis vi bare har én setning)

```
if (pris > 1500)
     System.out.println(”For dyrt”);
```

- Vi kan legge if-setninger inni if-setninger:

```
if (lonn < 1000000)
  if (ferieuker < 18)
     System.out.println(”Looser!”);
```


Eksempel: Body Mass Index

Body Mass Index (BMI) er et mål som kan regnes ut fra høyden og vekten til en person:

BMI	Vektstatus
Under 18.5	Undervekt
18.5 – 24.9	Normal vekt
25.0 – 29.9	Overvekt
30.0 eller høyere	Fedme

Beregn BMI ut fra høyde og vekt og gi melding om vektstatus!

Inndata og utdata

- **Inndata:**
  - Personens **høyde** (i m)
  - Personens **vekt** (i kg)
  - Leses fra terminal
- **Utdata:**
  - **BMI**
  - Skrives ut på skjerm, sammen med en av beskjedene
    - ***Undervekt*** hvis BMI < 18.5
    - ***Normal vekt*** hvis 18.5 <= BMI < 25
    - ***Overvekt*** hvis 25 <= BMI 30
    - ***Fedme*** hvis BMI >= 30

Transformere inndata til utdata

- Vi må kjenne formelen for å regne ut BMI. La

  vekt = personens vekt i kg

  hoyde = personens høyde i m

- Da er

  BMI = vekt / (hoyde*hoyde)


```
import easyIO.*;
class BodyMassIndex {
  public static void main (String[] args) {
    In tast = new In();
    System.out.print("Vekt (i kg): ");
    double vekt = tast.inDouble();
    System.out.print("Høyde (i cm): ");
    double hoyde = tast.inDouble()/100;
    double bmi = vekt / (hoyde * hoyde);
    System.out.println("BMI = " + bmi);

    if (bmi < 18.5)
      System.out.println(”Undervekt");
    else if (bmi < 25) {
      System.out.println(”Normalvekt");
    else if (bmi < 30) {
      System.out.println(”Overvekt");
    else
      System.out.println(”Fedme");
    }
}
```

```
Vekt (i kg): 100
Høyde (i cm): 170
BMI = 34.602076124567475:
Fedme
Vekt (i kg): 100
Høyde (i cm): 250
BMI = 16.0:
Undervekt
Vekt (i kg): 100
Høyde (i cm): 210
BMI = 22.675736961451246:
Normalvekt
```

while-løkker

- Vi kan utføre en blokk med setninger flere ganger :

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

- Hvis det logiske uttrykket er sant, utføres setningene i while-løkka
- Dette gjentas inntil det logiske uttrykket er usant. Da avsluttes løkka.

```
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);


}
```

```
$ 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
$
```

```
class LokkeTest {
  public static void main (String [] args){
        int k = 3;
        while (k > 0) {
            System.out.print("Nå er k = ");
            System.out.println(k);
            k = k - 1;
        }
  }
}
```

```
$ javac LokkeTest.java
$ java LokkeTest
Nå er k = 3
Nå er k = 2
Nå er k = 1
$
```

```
class WhileIJ {
  public static void main (String [] args) {
    int i = 1;
    int j = 6;
    while (i < j) {
      System.out.println("i = "+ i);
      System.out.println("j = "+ j);
      System.out.println();
      i = i + 1;
      j = j - 1;
    }

    System.out.println("i = "+ i);
    System.out.println("j = "+ j);
  }
}
```

```
$ javac
  WhileIJ.java
$ java WhileIJ
i = 1
j = 6

i = 2
j = 5

i = 3
j = 4

i = 4
j = 3
$
```

UNIVERSITETET I OSLO

Eksempel – Innlesning med sjekk

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

```
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);
    }
}
```

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

Evig løkke

UNIVERSITETET I OSLO

```
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(
                  "i=" + i + ", j=" + j );
            i++;
            j++;
        }
    }
}
```

Evig løkke - Kjøring

UNIVERSITETET I OSLO

- 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

. . .
. . .

i=82155, j=82156
i=82156, j=82157
i=82157, j=82158
i=82158, j=82159
i=82159, j=82160
i=82160, j=82161
i=82161, j=82162
i=82162, j=82163
i=82163, j=82164

. . .
. . .
```


UNIVERSITETET I OSLO

Litt mer vrient eksempel

- Du har 100 000 kr i banken
- Rentesatsen er 5%
- Årlig gebyr til banken på 370 kr trekkes ved slutt på år
- Gebyret øker med rentesatsen

Hvor mange år tar det for at beløpet har doblet seg?

- Løsning:
  - Bruk en while-løkke og beregn saldo år for år
  - Gå ut av løkka når saldo er dobbelt inngangsbeløp
  - Tell antall gjennomløp løkka har hatt

```
class Dobling{

    public static void main(String[] args){
       double startBelop = 100000;
       double rentefaktor = 1.05;
       double gebyr = 370;

       double belop = startBelop;
       int antallAar = 0;
       while( belop < 2*startBelop ){
           gebyr = gebyr*rentefaktor;
           belop = belop*rentefaktor - gebyr;
           antallAar++;
       }
       System.out.print("Antall år: ");
       System.out.println(antallAar);
    }
}
```

```
$ javac Dobling.java
$ java Dobling
Antall år: 16
$
```


Variant av while – do-while

```
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

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



```
class ForLokkeHvordan {
   public static void main (String [] args) {
        for( int i=0; i<5; i++) {

          System.out.println("Nå er i " + i);
        }
    }
}
```

initialisering

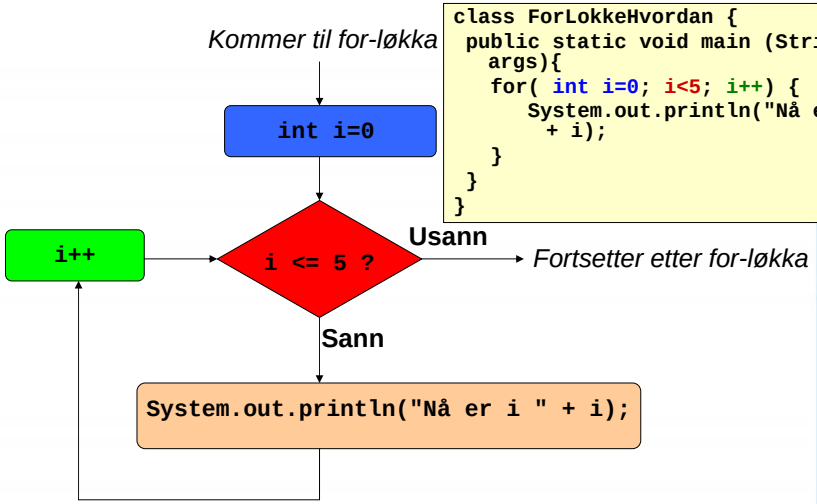
betingelse

oppdatering

i++ betyr
i = i + 1

```
$ java ForLokkeHvordan
Nå er i 0
Nå er i 1
Nå er i 2
Nå er i 3
Nå er i 4
$
```

Hvordan for-løkka virker




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


Nesting av løkker

- Husk å bruke forskjellig ”tellevariabel” i de forskjellige løkkene når de legges inne i hverandre!
- I 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);
            }
        }
    }
}
```

```
1*1=1
1*2=2
1*3=3
1*4=4
1*5=5
1*6=6
1*7=7
1*8=8
1*9=9
1*10=10
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
2*10=20
3*1=3
3*2=6

...
```

Heltallsdivsjon

- Gitt en *divisor* **d** kan skrive et heltall **a** som:

$$a = f * d + r$$

  - **r** < **d** kalles *resten modulo* ***d***
  - **f** kalles *faktoren modulo* ***d***
  - Divisjonsoperatoren (/) gir faktoren
  - Modula-operatoren (%) gir resten

```
int a  = 13
int d  = 5
int f  = a / d;  // Nå er f lik 2
int r  = a % d;  // Nå er r lik 3
```

- Eksempel: 13 = 2 * 5 + 3

Divisjon og rest modulo 12

- Tenk deg at du skal “telle rundt klokka”
  - Gitt en urskive som viser 12 timer.
  - Vi starter øverst og sier at klokka da er 0
- Når du teller 14 timer:
  - du har gått rundt klokka 1 gang
  - klokka viser 2
- Du har nå telt 14 modulo 12:
  - 14 / 12 er antall ganger du har gått rundt urskiven (dvs 1)
  - 14 % 12 er det tallet som urviseret står på (dvs 2)

UNIVERSITETET I OSLO

Er N et primtall?

- Et primtall er et tall større enn 1 som bare er delelig med 1 og seg selv
- N er delelig med d hvis og bare hvis (N%d == 0)
- For å finne om N er primtall:
  - Sjekk resten ved divisjon med 2, … , N-1
  - Hvis ingen av disse er 0, er N et primtall
  - Hvis noen av disse er 0, er N ikke primtall

d	25%d	primtall
2	1	true
3	1	true
4	1	true
5	0	false
6	1	false
7	4	false
8	1	false
9	7	false
10	5	false
11	3	false
12	1	false
13	12	false
14	11	false
15	10	false
16	9	false
17	8	false
18	7	false
19	6	false
20	5	false
21	4	false
22	3	false
23	2	false
24	1	false

Alle gjennomløp av løkka for N = 25

Merk:
Vi kan hoppe ut av løkka etter 4 steg: Da vet vi at 25 ikke er primtall! Det kan vi oppnå på to måter:
- Test for dette i betingelsen i for-løkka (se neste slide)
- Skriv flg. setning i if-blokken inne for-løkka: break;

```
boolean primtall = true;
for(int d=2; d<N; d++)
  if( N%d == 0 ){
     primtall = false;
  }
// Her har primtall riktig verdi
```

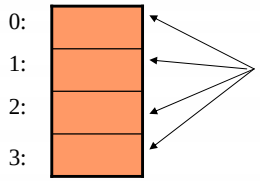
```
class SjekkOmPrimtall {
  public static void main(String[] args){
      final int N = 17333;
      boolean primtall = true;
      for(int d=2; primtall && d<N; d++)
         if( N%d == 0 ){
             primtall = false;
         }
      System.out.print( N + " er ");
      if( !primtall ) System.out.print("ikke ");
      System.out.println("primtall");
  }
}
```

17333 er primtall
17334 er ikke primtall

Arrayer

UNIVERSITETET I OSLO

- En variabel kan holde en enkelt verdi:
  - en int-variabel har plass til ett heltall
  - en String-variabel har plass til en enkelt tekststreng
- Arrayer kan holde på mange verdier:
  - en int-array har plass til mange heltall
  - en String-array har plass til mange tekststrenger
- Verdiene i en array med lengde N har hver sin *indeks*:
  - 0, 1, 2, . . . , N-1
- En array tlf med lengde 4 kan tegnes slik:



Hver av disse boksene har plass til én verdi

Deklarere og opprette arrayer

- Deklarere en array:
  `<datatype>[] arrayNavn;`
- Opprette en array:
  `arrayNavn = new <datatype>[N];`
- Deklarere og opprette i en operasjon:
  `<datatype>[] arrayNavn = new <datatype>[N];`

```
int[] tlf = new int[4];
double[] mmRegn = new double[100];
String[] kontakter = new String[1000];
```


Verdiene i en array

- Deklarasjon og oppretting av array:
  `int[] tlf = new int[4];`
- Navn på de enkelte verdiene i arrayen:
  `tlf[0], tlf[1], tlf[2], tlf[3]`
- Lengden på arrayen (her: 4) fås slik:
  `tlf.length // NB: ingen parenteser`

Finne den yngste av flere

```
import easyIO.*;
class FinnDenYngste {
    public static void main (String [] args) {
        In tast = new In();
        System.out.print(
            "Hvor mange personer? ");
        int antall = tast.inInt();

       String[] navn = new String[antall];
       int[] alder = new int[antall];

        for (int i=0; i<antall; i++) {
            System.out.print("Navn: ");
            navn[i] = tast.inLine();
            System.out.print("Alder: ");
            alder[i] = tast.inInt();
        }
```

Leser inn antall personer

Oppretter arrayer

Leser inn navn og alder for alle


```
// . . .

        int minPos = 0;

        for (int i=1; i<antall; i++)
            if (alder[i] < alder[minPos];)
                minPos = i;

        System.out.println("Den yngste er " +
            navn[minPos] + " som er " +
            alder[minPos] + " år");
    }
}
```

Posisjonen til den yngste vi kjenner i arrayene. Start med den første!

Sjekker om vi har funnet en yngre …

Går så gjennom hver av de andre posisjonene i i arrayene

…og endrer i så fall posisjonen til den yngste vi kjenner til nå

```
$ javac FinnDenYngste.java
$ java FinnDenYngste
Hvor mange personer? 3
Navn: Per
Alder: 41
Navn: Kari
Alder: 40
Navn: Arne
Alder: 60
Den yngste er Kari som er 40 år
$
```


Alle primtall opp til MAX

- Lag en boolean array 0, … , MAX:

```
boolean[] primtall = new boolean[MAX+1];
```

  - For hver d fra 2 til MAX-1
    - For hvert tall fra d+1 til MAX

      Hvis tall er delelig på d, er ikke tall primtall

```
for(int d=2; d<MAX; d++)
  for(int tall=d+1; tall<=MAX; tall++)
    if( tall%d == 0 )
       primtall[tall] = false;
```

Forbedringer

- Hvis vi har sjekket at et tall er delelig med 2, trenger vi ikke også å sjekke at det er delelig med 4 osv
- Hvis vi vet at tall et delelig med et noe, trenger vi ikke sjekke om igjen med noe annet

```
for(int d=2; d<MAX; d++)
  if( primtall[d] )
    for(int tall=d+1; tall<=MAX; tall++)
     if(primtall[tall] && tall%d == 0 )
       primtall[tall] = false;
```

d	tall	tall%d	primtall[tall]
2	3	1	true
2	4	0	false
2	5	1	true
2	6	0	false
2	7	1	true
2	8	0	false
2	9	1	true
2	10	0	false
3	5	2	true
3	7	1	true
3	9	0	false
5	7	2	true

Alle gjennomløp av indre løkke for MAX = 10

```
class Primtall {
    public static void main(String[] args){
        final int MAX = 100;
        boolean[] primtall = new boolean[MAX+1];
        // Gi riktig startverdi. NB! 0 og 1 ikke prim
        for(int i=2; i<=MAX; i++)
            primtall[i] = true;

        for(int d=2; d<MAX; d++)
           if( primtall[d] )
              for(int tall=d+1; tall<=MAX; tall++)
                if( primtall[tall] && tall%d == 0 )
                    primtall[tall] = false;

        for(int i=0; i<=MAX; i++)
            if( primtall[i] ) System.out.println(i);
    }
}
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```