



INF1000 – Uke 4

Forgreininger, løkker og arrayer

1



Oversikt

- Litt repetisjon
- Blokker og forgreininger
 - if-setninger
 - if-else-setninger
 - switch-setninger
- Løkker
 - while-løkker
 - do-while-løkker
 - for-løkker
- Arrayer
 - Opprette, skrive, lese og lete i
 - Flere dimensjoner

2



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.

3



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();
```

4

Eksempel

Vi importerer pakken easyIO.

```
import easyIO.*;
```

Vi oppretter en verktøykasse for lesing fra terminal og lager en variabel tast som blir vårt håndtak til denne verktøykassen.

```
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);  
    }  
}
```

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

5

Resultat

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

Sammenlikning – String

- En String er ikke en basistype, men et objekt.
 - Mer om objekter siden.
- Vi må bruke en egen funksjon for å sammenlikne en String med en annen

```
enString.equals(enAnnenString);
```

- For eksempel
"ja".equals(svar);

7

Sammenlikning – String

```
class SammenlikneTekst {  
    public static void main (String[] args) {  
        String ikkeNoe;  
        String noe = "noe";  
  
        boolean erSann = "noe".equals(noe);  
  
        boolean erIkkeSann = "noeannet".equals(noe);  
  
        //boolean girFeil1 = noe.equals(ikkeNoe);  
  
        //boolean girFeil2 = ikkeNoe.equals("noeannet");  
  
        System.out.println("erSann=" + erSann);  
        System.out.println("erIkkeSann=" + erIkkeSann);  
    }  
}
```

Er en String et objekt?

8

Blokker

- En **programblokk** er en samling med programsetninger omsluttet av krøllparenteser
- Setningene i main-metoden ligger for eksempel inne i en blokk
- Blokker kan **nøstes** inne i hverandre, slik at vi kan ha blokker inne i blokker
- En variabel som er deklartert inne i en blokk er kun definert ("synlig") fra stedet den er deklartert til slutten av blokken. Vi kaller det **skopet** til variabelen.

9

Skop – Lovlig eksempel

```
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(k);
    }
}
```

10

Skop – Ikke lovlig eksempel

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

11

Skop – Ikke lovlig eksempel

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

12

Programmer med forgreninger

- En svært nyttig programmeringsteknikk er å bruke forgreninger, dvs forskjellige instruksjoner utføres i ulike situasjoner.

- Vi kan få til dette med en **if-setning** (pseudokode):

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

et uttrykk som enten er true eller false, 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)

- Eksempel:

```
if (x > 0) {
    System.out.println("Tallet er positivt");
} else {
    System.out.println("Tallet er ikke positivt");
}
```

13

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:

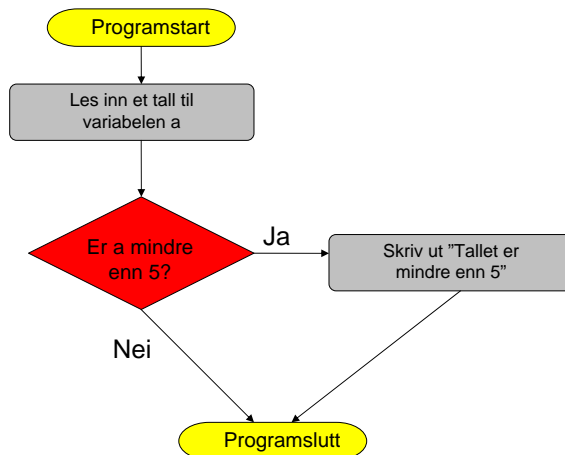
```
if (lønn < 400000) {
    if (ferieuker < 8) {
        System.out.println("Ikke søk på jobben");
    }
}
```

- Vi kan lage sammensatte if-setninger:

```
if (a < 10) { // a ikke er positivt heltall
    System.out.println("Ett siffer");
} else if (a < 100) {
    System.out.println("To siffer");
} else {
    System.out.println("Mer enn to siffer");
}
```

14

Flytdiagram



15

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");
        }
    }
}
```

16

Eksempel: Body Mass Index

Oppgave:

Body Mass Index (BMI) er et mål som kan regnes ut fra høyden og vekten til en person. Ifølge verdens helseorganisasjon (WHO)¹ :

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

Vi skal lage et program som beregner BMI ut fra høyde og vekt og gir melding om hvilken vektstatus (se tabellen) det tilsvarer.

¹Se http://www.who.int/hpr/NPH/docs/gs_obesity.pdf

Hvordan løse oppgaver

1. **Bestem programmets oppførsel sett 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. **Avgjør hvordan du skal transformere inndata til utdata:**
 1. Hvordan skal inn- og utdata representeres (lagres) i programmet?
 2. Reduser transformasjonen inndata -> utdata til en sekvens av trinn hvor hvert trinn gjør en enkel ting med dataene og hvor hvert trinn er enkelt å programmere.
3. **Skriv programkode (og test løsningen).**

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 <= 18.4)
 - **Normal vekt** (hvis 18.5 <= BMI <= 24.9)
 - **Overvekt** (hvis 25.0 <= BMI <= 29.9)
 - **Fedme** (hvis BMI >= 30.0)

Transformere inndata til utdata

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

$$\text{vekt} = \text{personens vekt i kg}$$

$$\text{hoyde} = \text{personens høyde i m}$$

- Da er

$$\text{BMI} = \text{vekt} / (\text{hoyde} * \text{hoyde})$$

Ferdig program

```
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 høyde = tast.inDouble()/100;
        double bmi = vekt / (høyde * høyde);
        System.out.println("BMI = " + bmi);

        if (bmi <= 18.4) {
            System.out.println("Vektstatus: undervekt");
        } else if (bmi <= 24.9) {
            System.out.println("Vektstatus: normalvekt");
        } else if (bmi <= 29.9) {
            System.out.println("Vektstatus: overvekt");
        } else {
            System.out.println("Vektstatus: fedme");
        }
    }
}
```

21

Alternativ til if-else: switch

- En sammensetning av flere if-setninger kan i noen tilfeller erstattes med en switch-setning:

```
switch (uttrykk) {
    case verdil:
        <instruksjoner>
        break;
    ....
    case verdiN:
        <instruksjoner>
        break;
    default:
        <instruksjoner>
}
```

22

Alternativ til if-else – switch

- <uttrykk> må være av typene **int**, **byte**, **short** eller **char**
- Verdiene <verdi 1>, <verdi 2>, ... <verdi n> må være **konstanter**
 - Konstanter er verdier som ikke kan endres. Altså kan de ikke være variabler eller uttrykk
- Uttrykket regnes ut og utførelsen fortsetter ved den verdien som er lik resultatet
- Det letes etter et treff ovenfra og nedover.
- Når det finnes et treff utføres setningene frem til **break**: Etter break er man ferdig med switch-setningen
 - Vær obs på at dersom det ikke står **break** vil utførelsen fortsette gjennom neste **case**
 - Bruk alltid(?) **break** med mindre det finnes en virkelig god grunn
- Dersom det ikke finnes noen match vil setningene etter **default** utføres
- Dersom man forsøker å bruke samme verdi flere ganger gir det kompilingsfeil

23

Eksempel

```
class BrukAvSwitch {
    public static void main (String [] args) {
        char c = 'x';
        switch(c) {
            case 'a':
                System.out.println("Tegnet var en a");
                break;
            case 'b':
                System.out.println("Tegnet var en b");
                break;
            default :
                System.out.println(
                    "Tegnet var ikke a eller b");
        }
    }
}
```

24

Oppgave 1 Hva skriver programmet ut?

```
class IfTest {
    public static void main (String [] args) {
        double x = -0.5;
        double y = 0.5;
        if (1/2 > 0) {
            System.out.println("A");
        }
        if ((int) x == (int) y) {
            System.out.println("B");
        }
        if (x < y) {
            if (x < 0) {
                if (y < 0) {
                    System.out.println("C");
                }
            } else {
                System.out.println("D");
            }
        }
    }
}
```

25

Oppgave 1

```
$ javac IfTest.java
$ java IfTest
B
```

26

Oppgave 3

- Hva skriver programmet ut hvis input er 2?
- Hva skriver programmet ut hvis input er 4?

```
import easyIO.*;
class SwitchFallThrough {
    public static void main (String [] args) {
        In skrivHer = new In();
        int x = skrivHer.inInt();
        switch(x) {
            case 1:
                System.out.println("Tallet er lik 1");
            case 2:
                System.out.println("Tallet er mindre eller lik 2");
            case 3:
                System.out.println("Tallet er mindre eller lik 3");
                // Denne gjør at default ikke utføres etter 3.
                break;
            default :
                System.out.println("Tallet er mindre enn 3");
        }
    }
}
```

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

28

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);
    }
}
```

29

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
$
```

30

Oppgave 4

Hva skriver programmet ut?

```
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;
        }
    }
}
```

31

Kompilering og kjøring

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

32

Oppgave 5

Hva skriver programmet ut?

```
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);
    }
}
```

33

Kompilering og kjøring

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

i = 2
j = 5

i = 3
j = 4

i = 4
j = 3
$
```

34

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.

35

Oppgave 5

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);
    }
}
```

36

Kompilering og kjøring

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

37

Evig løkke

- Dersom testen i while-løkken **aldri blir usann** (false), vil utførelsen av while-løkken 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
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)
. . .
. . .
. . .
```

39

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.

40

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;>  
}
```

41

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);  
        }  
    }  
}
```

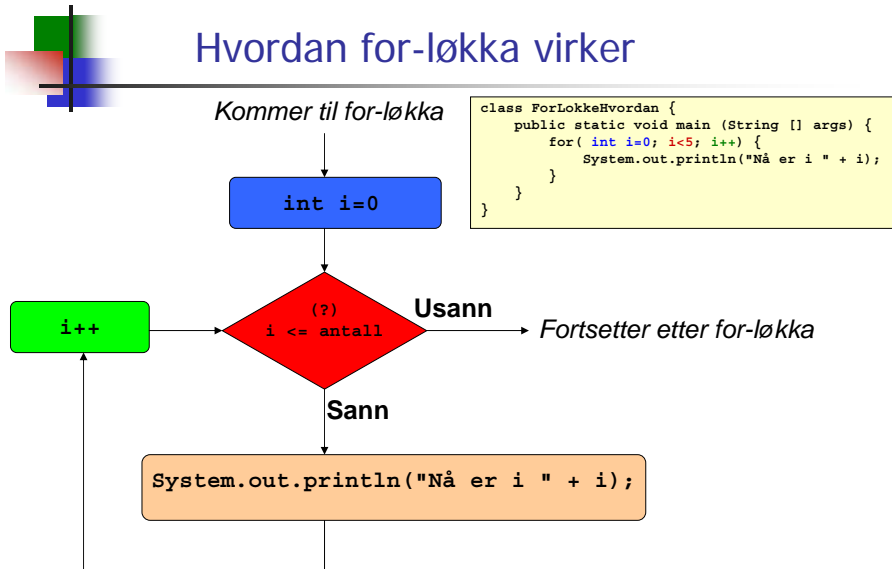
Initialisering

Oppdatering

Betingelse

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

Hvordan for-løkka virker



43

Nesting av løkker

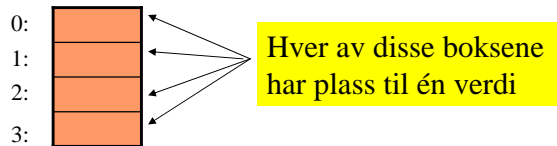
- Det er ofte behov for å neste løkke-setninger inne i hverandre og vi kommer til å se mange eksempler på det etter hvert.
- Man må passe på å bruke forskjellig "tellevariabel" i de forskjellige løkkene.
- I eksemplet er **i** og **j** brukt.

```
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);  
            }  
        }  
    }  
}
```

44

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, . . . , N-1 hvor N = lengden til arrayen
- En array x med lengde 4 kan tegnes slik:



45

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];`
- 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];`

46

Verdiene i en array

- Anta at vi har deklart 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

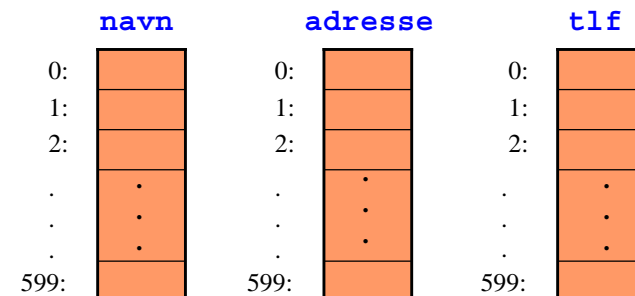
47

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

- Resultatet kan visualiseres (tegnes) slik



48

Eksempel: lese og skrive ut

- Program som leser data om et antall personer fra input.

```
import easyIO.*;
class LesInnPersoner {
    public static void main (String [] args) {
        In tast = new In();
        String[] navn = new String[3];
        for (int i=0; i<navn.length; i++) {
            System.out.print("Navn: ");
            navn[i] = tast.inLine();
        }
        for (int i=0; i<navn.length; i++) {
            System.out.println(navn[i]);
        }
    }
}
```

Oppretter array

Legger inn verdi

Bruker lengden i betingelsen

Leser ut verdi

49

Automatisk initialisering av arrayer

- Når en array blir opprettet, blir den automatisk initialisert (dvs verdiene er ikke udefinerte når arrayen 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] = "";`.
- Generelt, når vi bruker `new`, får vi 'null-fylt' det vi lager med `new`. (mye mer bruk av `new` senere)

50

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"};
```

51

Eksempel – Finne den yngste

```
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 navn og alder i disse.

52

Eksempel – fortsetter

```
// . . .
```

```
int minste = alder[0];  
int minPos = 0;
```

Skal hele tiden legge den minste her. Starter med den første

Posisjonen til den minste i arrayen

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

Sjekker om vi har funnet en som er mindre og oppdaterer i så fall verdiene.

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

Eksempel – Finne den yngste

```
$ javac FinnDenYngste.java  
$ java FinnDenYngste  
Hvor mange personer? 2  
Navn: Arild  
Alder: 40  
Navn: Arne  
Alder: 60  
Den yngste er Arild som er 40 år  
$
```

En array-variabel er en adresse (en peker)

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

```
int[] printtall = {2, 3, 5, 7, 11, 13};
```
- kan visualiseres slik:



Oppgave

- Hva skriver programmet ut?

```
class ToArrayer {  
    public static void main (String [] args) {  
        int[] x = new int[5];  
        int[] y = x;  
  
        for (int i=0; i<x.length; i++) {  
            x[i] = 10 + i;  
        }  
  
        for (int i=0; i < y.length; i++) {  
            System.out.println(y[i]);  
        }  
    }  
}
```

Resultat

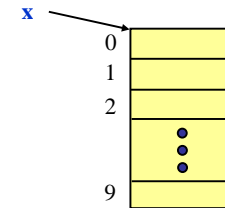
```
$ javac ToArrayer.java
$ java ToArrayer
10
11
12
13
14
$
```

Hva skjedde?

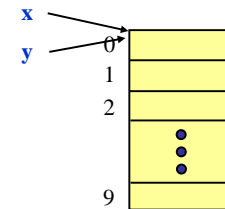
Etter å ha utført instruksjonen

.... så er situasjonen denne:

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



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



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];
}
```

2 dimensjonale arrayer (2D)

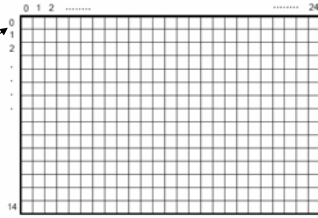
- Vi kan også deklarere todimensjonale (og høyere-dimensjonale) arrayer.
- Eksempel:

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

To-dimensjonale (2D) arrayer

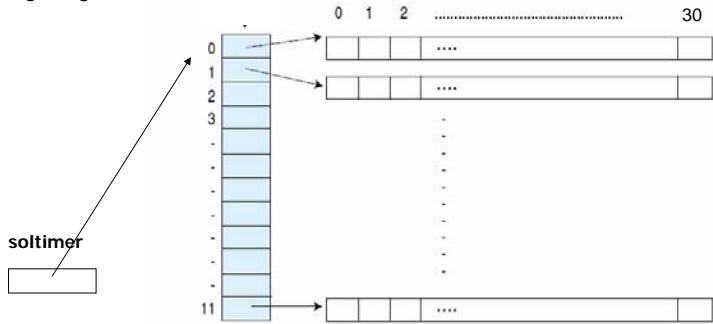
- slik tenker vi oss det
- og slik er det

oljefelter



```
int[][] soltimer = new int[12][31];
```

gir følgende:



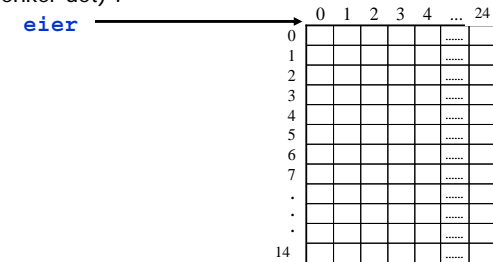
Figur 5.3 En todimensjonal array med arraypekere og arrayobjekter.

Flerdimensjonale arrayer

Eksempel:

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

- Resultat (slik vi tenker det) :



- Eksempler på lovlige operasjoner:

```
eier[3][4] = "Petrol A/S";
```

```
int antallRader = eier.length; // 15
```

```
int antallKolonner = eier[0].length; // 25
```