

INF1010

Grafisk brukergrensesnitt med Swing og awt del 1



GUI (Graphical User Interface)-programmering

- Hvordan lage et vindu på skjermen
 - Hvordan legge ulike komponenter i vinduet (trykknapper, tekstfelter, tekst, bilder, mm) Grafikk (tegning i vinduet)
 - Kort om layout (utlegg? utseende?) av vinduer
- Litt om hvordan Java-programmet vårt fanger opp knappetrykk
 - Enklest mulig eksempel
- Et enkelt Model-Utsyn-Kontroll-eksempel
- Grafikk (tegning i vinduet)
- Grundigere om hendelsesmodellen og oppfanging av tastaturtrykk, musetrykk, musebevegelser.
- Mer om hvordan få data inn fra brukeren via vinduer (Brukeren trykker på knapper, fyller ut data,...)
 - Hvordan får programmet tak i disse data?
- "Listener-e" vs. "Adapter-e"



Om Grafiske Bruker-Grensesnitt (GUI)

- Data inn og ut i DOS-/kommandovinduet/shell ofte ikke naturlig.
- GUI: Vi dekker bare litt, men nok til å gå videre selv. Javas klassebibliotek for GUI har mer enn 300 metoder, men bare et lite antall av disse nyttes i praksis.
- Ofte tar vi utgangspunkt i et eksempel som virker og utvider / innskrenker dette.
- Når man jobber i industrien, bruker man ofte verktøy for 'drag-and-drop' konstruksjon av GUI. Dere skal lære GUI fra grunnen av og løse oppgavene med **Swing** og **awt** (advanced window toolkit).
- Java system består av et litt eldre system **awt**, og et litt nyere system **swing** som er bygget på **awt**. Men ikke alt er skrevet om, så vi trenger i nesten alle programmer å importere begge (-merk javax):

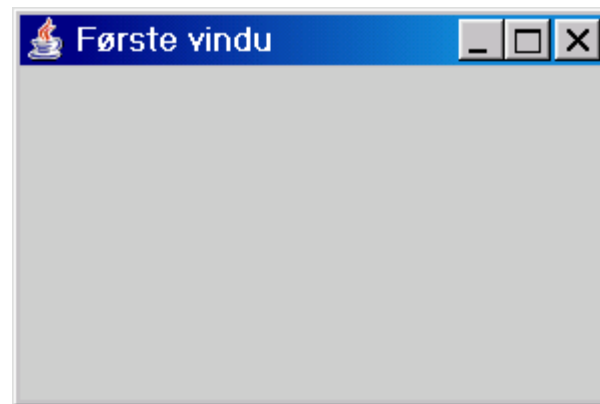
```
import javax.swing.*;  
import java.awt.*;
```

(De klasser som begynner på J er swing, resten er awt.)



Hvordan gjør vi det: To måter

- Klassen JFrame lager vinduer.
- Kan enten bruke JFrame som den er, eller
- Lage en subklasse av JFrame og legge til den spesielle koden vi ønsker i subklassen.
- Eksempel:



Bruke klassen JFrame som den er

(komponentmetoden)

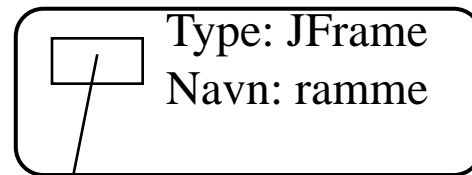
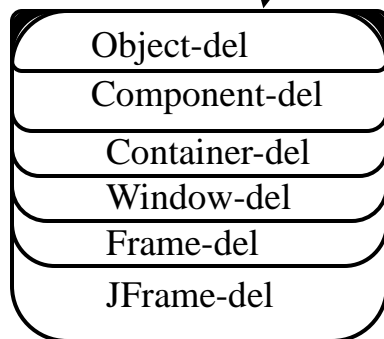
```
import javax.swing.*;
import java.awt.*;

class Gui01 {
    public static void main(String [] a) {
        new Testeksempel();
    }
}

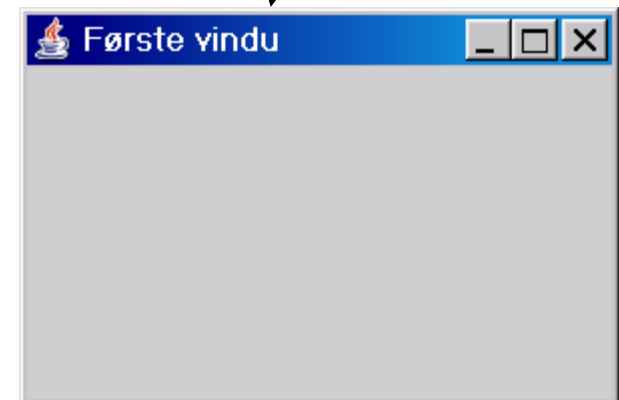
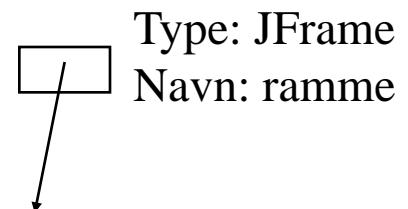
class Testeksempel{

    Testeksempel(){
        JFrame ramme;
        ramme = new JFrame("Første_vindu");
        ramme.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ramme.setSize(300, 200);
        ramme.setVisible(true);
    }
}
```

Pluss mange
Implementasjoner
av forskjellige
grensesnitt



Objekt av
klassen
JFrame
← i minne
på skjermen →

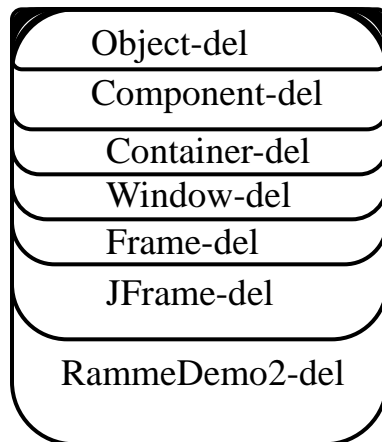


Subklasse av JFrame (mest vanlig)

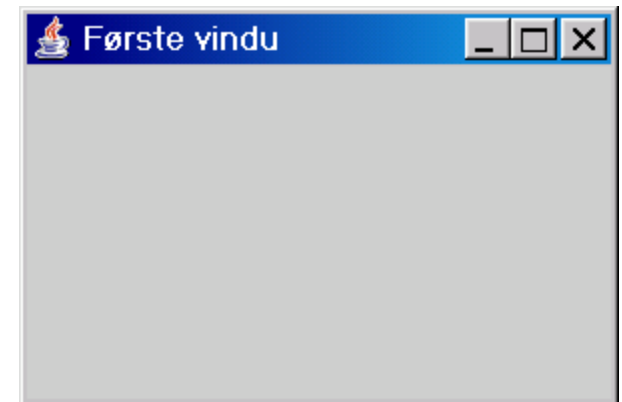
```
import javax.swing.*;  
import java.awt.*;
```

(subklassemetoden)

```
class RammeDemo2 extends JFrame {  
    RammeDemo2() {  
        super("Første vindu"); // En annen måte å sette tittel på rammen  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        setSize(300,200);  
        setVisible(true);  
    }  
    public static void main(String[] args) {  
        new RammeDemo2();  
    }  
}
```



Objekt av
klassen
RammeDemo2
← i minne
på skjermen →



Standard avslutning

- Dette bør med i alle vinduer

`setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`

Setter x i  til å virke vanlig *

`setVisible(true);`

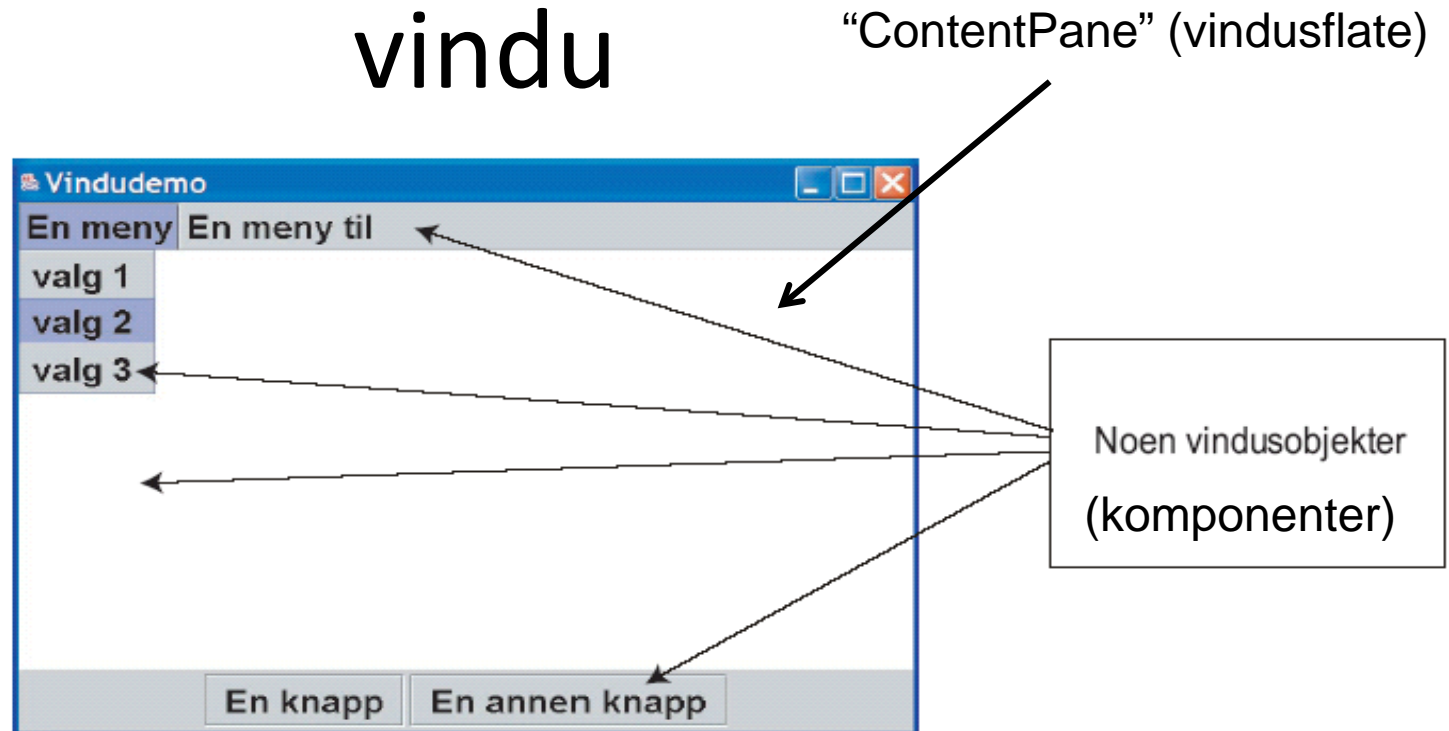
er nødvendig for at vinduet skal vise seg fram på skjermen

Først lager vi vinduet med alle dens felter, trykknapper,..osv, så viser vi det fram

* Prøv uten og se hva som skjer



Det er mange komponenter i et vindu



Figur 15.2 Alle komponenter i et vindu representeres av et objekt

Alle komponenter legges på “vindusflaten”

Hva gjør vi når vi lager et vindu – forenklet versjon (flere punkter for mer kompliserte vinduer senere).

1. Vi lager et objekt for vinduet , subklasse av JFrame – og setter navn på rammen
2. Kan sette størrelsen
`setSize(300, 200);`
3. Får tak i en peker til vindusflaten
`Container lerret = getContentPane();`
4. Lager objekter for alle de komponentene vi vil ha i vinduet og legger alle disse inn i vindusflaten
`lerret.add(..<peker til et objekt for en komponent >,.....)`
5. Setter inn at avslutningsknappen skal virke:
`setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
6. Sier fra at vinduet skal vise seg fram
`setVisible(true);`



Noen komponenter vi kan legge inn

JButton:

`JButton knapp = new JButton("Trykk her");` *En knapp som brukeren kan trykke på. Parameteren i konstruktøren angir teksten på knappen.*

JLabel:

`JLabel etikett = new JLabel("Skriv inn navn");` *En etikett som kan inneholde enten tekst eller bilder. Brukes ofte som ledetekst til JText-Field.*

JTextField:

`JTextField tekstfelt = new JTextField(30);` *Et tekstfelt hvor brukeren kan skrive inn tekst. Metoden `getText()` returnerer teksten i feltet.*

JTextArea:

`JTextArea tekstvindu = new JTextArea(10, 30);` *Et tekstvindu hvor programmet kan vise fram tekst. Metoden `getText()` returnerer teksten i vinduet.*

JScrollPane:

`JScrollPane rulleVindu = new JScrollPane(tekstvindu);` *Lager horisontale og vertikale rullefelt rundt et element. Brukes ofte i sammenheng med JTextArea som legges inn i rullevinduet. Parameteren er elementet vi ønsker lagt inn i rullevinduet.*

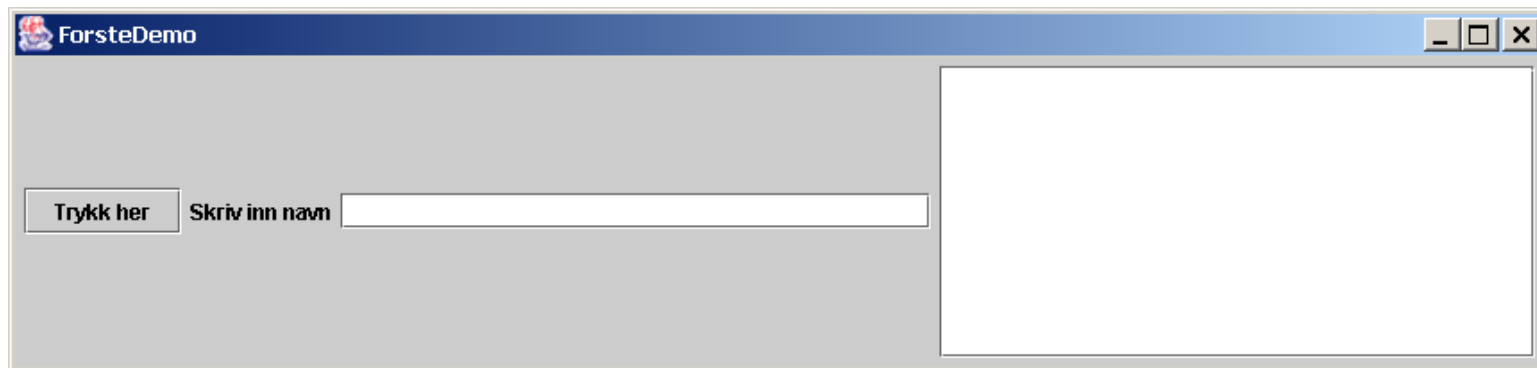
Jpanel:

`JPanel panel = new JPanel();` *Et "panel" som kan inneholde andre komponenter.*

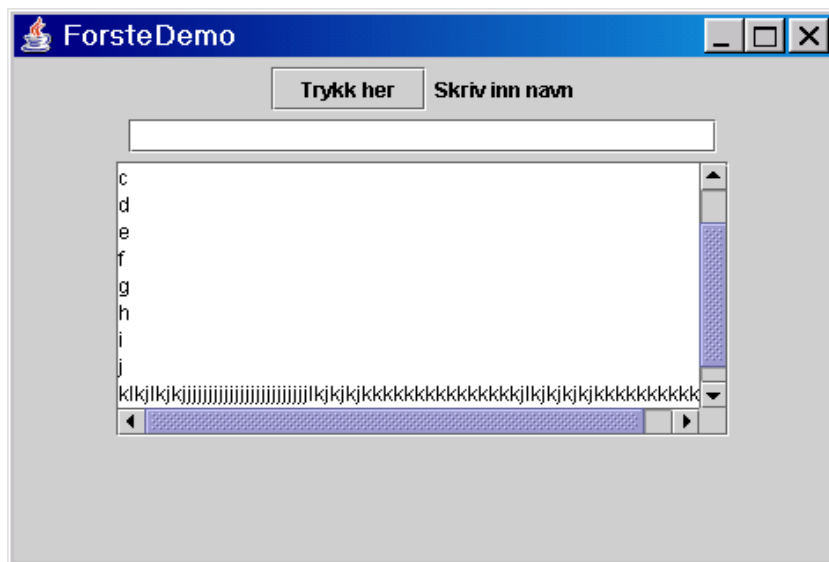


Vindu med komponenter (ikke så bra utseende enda)

Slik vil det blir når det kommer opp



Etter å ha dradd i det (laget mindre bredde). Rullefeltet framkommer når vi fyller tekstfeltet



Mer om å lage et **panel** og legge ting inn i det.

Deretter legges panelet inn i vindusflaten

// start og avslutning som før, dette er inne i konstruktøren

// Først lages elementene:

JBUTTON knapp = new JButton("Trykk her");

JLabel etikett = new JLabel("Skriv inn navn");

JTextField tekstfelt = new JTextField(30);

JTextArea tekstvindu = new JTextArea(10, 30);

JScrollPane rullevindu = new JScrollPane(tekstvindu);

// Lager et panel og legger elementene til dette.

JPanel panel = new JPanel();

panel.add(knapp);

panel.add(etikett);

panel.add(tekstfelt);

panel.add(rullevindu); // tekstvinduet er inne i rullevinduet

//Får tak i peker til vindues lerret og legger panelet inn

Container lerret = getContentPane(); // er inne i et JFrame-obj.

lerret.add(panel);



To viktige begreper

- **Container**

- Klasse(r) som kan inneholde komponenter og andre Containerer (som igjen kan inneholde...)
- JFrame har en innebygd Container som alt skal legges i som skal inn i vinduet, og vi får tak i den med:

`getContentPane() ;`

- Panel er en (subklasse av) Container

- **LayoutManager**

- Er klasser som automatisk sørger for at det vi legger inn (add ()) i en Container blir ordnet i en bestemt rekkefølge, og at plasseringen av komponentene blir OK hvis brukeren endrer størrelsen på vinduet.

Alle Containerer har en bestemt standard layoutManager
(hvis vi ikke endrer den)

- JPanel har FlowLayout (fra venstre mot høyre (i en (eller flere) rekke(r)))
- Den innebygde Containeren i JFrame har BorderLayout (fem felt: NORTH, WEST, SOUTH, EAST og CENTER)
 - eks: panel.add(knapp,BorderLayout.NORTH);
 - NB: Ulike LayoutMangere har add()-metoder med *ulikt* antall parametre. Husk å bruk dem riktige (ellers skjer ingen ting)



Layoutmanagers:

Layoutmanagere:

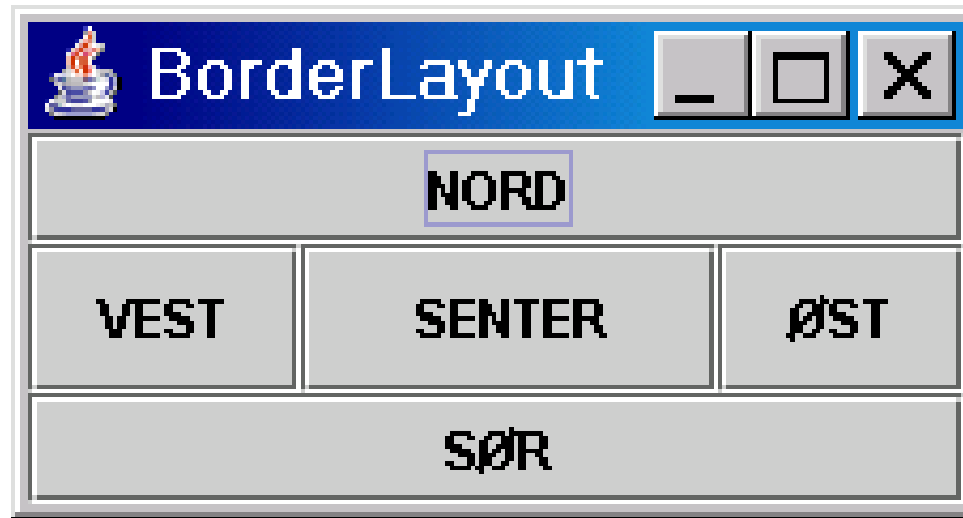
- Bordelayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- SpringLayout

<http://docs.oracle.com/javase/tutorial/uiswing/layout/layoutlist.html>

<http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>



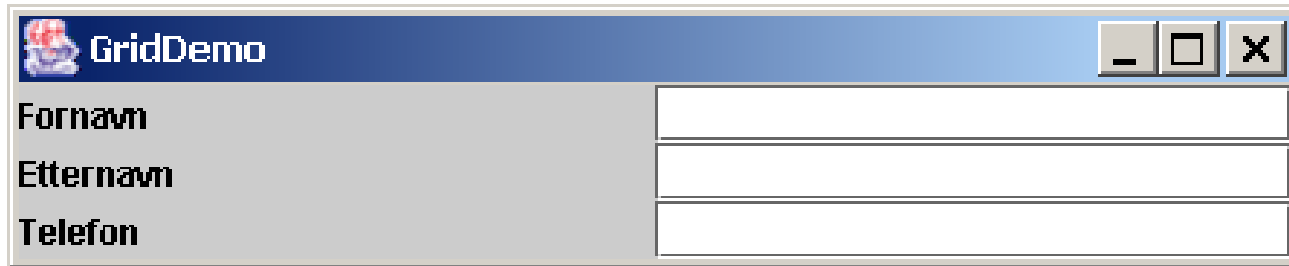
Bruk av BorderLayout (start og slutt som før)



```
Container lerret = getContentPane();  
lerret.setLayout(new BorderLayout());  
lerret.add(new JButton("NORD"), BorderLayout.NORTH);  
lerret.add(new JButton("SØR"), BorderLayout.SOUTH);  
lerret.add(new JButton("ØST"), BorderLayout.EAST);  
lerret.add(new JButton("VEST"), BorderLayout.WEST);  
lerret.add(new JButton("SENER"), BorderLayout.CENTER);
```



Grid Layout (start og slutt som før)



The screenshot shows a Java Swing window titled "GridDemo". The window has a standard Mac OS X-style title bar with a red close button, a yellow maximize button, and a green window button. The main content area is a light gray rectangle. It contains three labels ("Fornavn", "Etternavn", "Telefon") on the left, each followed by a white text input field on the right. The labels and input fields are arranged in a 3x2 grid.

```
Container lerret = getContentPane();
```

```
lerret.setLayout(new GridLayout(3, 2)); // 3 rader, 2 kolonner
```

```
lerret.add(new JLabel("Fornavn"));
```

```
lerret.add(new JTextField(20));
```

```
lerret.add(new JLabel("Etternavn"));
```

```
lerret.add(new JTextField(20));
```

```
lerret.add(new JLabel("Telefon"));
```

```
lerret.add(new JTextField(20));
```

Vi legger inn radvis i tabellen.

Å legge rett inn med **new** som her, virker ikke hvis vi vil ha input etterpå (fordi vi ikke har noen pekere til komponentene).



Vi endrer til FlowLayout (start og slutt som før)



```
Container lerret = getContentPane();  
lerret.setLayout(new FlowLayout());  
lerret.add(new JLabel("Fornavn"));  
lerret.add(new JTextField(20));  
lerret.add(new JLabel("Etternavn"));  
lerret.add(new JTextField(20));  
lerret.add(new JLabel("Telefon"));  
lerret.add(new JTextField(20));
```

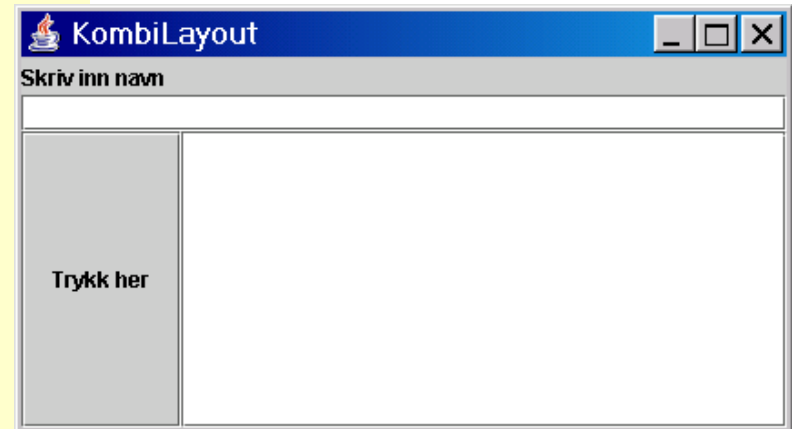
Vi legger inn etter hverandre .



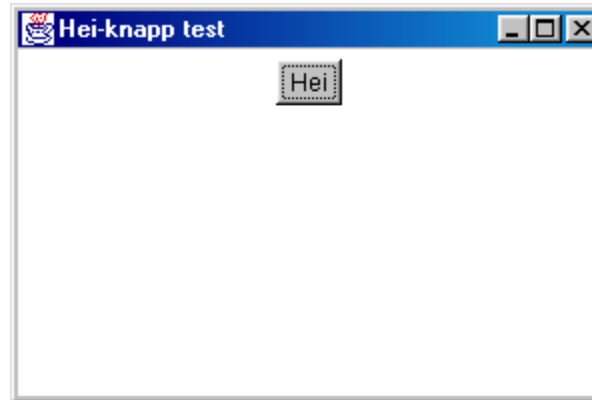
Kombiner flere layout i ett vindu

```
setTitle("KombiLayout");  
// Lager komponentene  
JButton knapp = new JButton("Trykk her");  
JLabel etikett = new JLabel("Skriv inn navn");  
JTextField tekstfelt = new JTextField(30);  
etikett.setLabelFor(tekstfelt);  
JTextArea tekstvindu = new JTextArea(10, 30);  
JScrollPane rullevindu = new JScrollPane(tekstvindu);  
// Benytter GridLayout.  
JPanel tekstPanel = new JPanel();  
tekstPanel.setLayout(new GridLayout(2, 1));  
tekstPanel.add(etikett);  
tekstPanel.add(tekstfelt);  
// Legger panelet og resten av komponentene i JFrame-en  
Container lerret = getContentPane();  
lerret.add(tekstPanel, BorderLayout.NORTH);  
lerret.add(rullevindu, BorderLayout.CENTER);  
lerret.add(knapp, BorderLayout.WEST);
```

- Lag et eller flere JPanel, gi dem hver sin layout
- Legg komponentene til i de ulike panelene, evt. også rett i lerret (add)
- Legg JPanelene inn i lerret (add)



En knapp med reaksjon



Vi skal lage det aller enkleste programmet vi kan tenke oss med én knapp som reagerer på at vi trykker på den ved å gi en utskrift i kommando-vinduet:

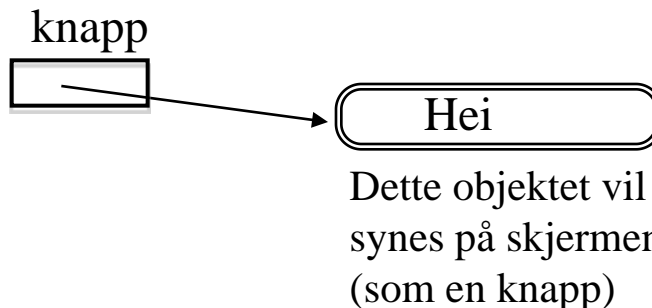
```
C:\javaprogram> javac Vindu.java
C:\javaprogram> java Vindu
noen sa hei til meg
noen sa hei til meg
noen sa hei til meg
```



Hvordan lage en knapp som lytter

```
 JButton knapp;
```

```
 knapp= new JButton("Hei");
```



```
 getContentPane().add(knapp);
```

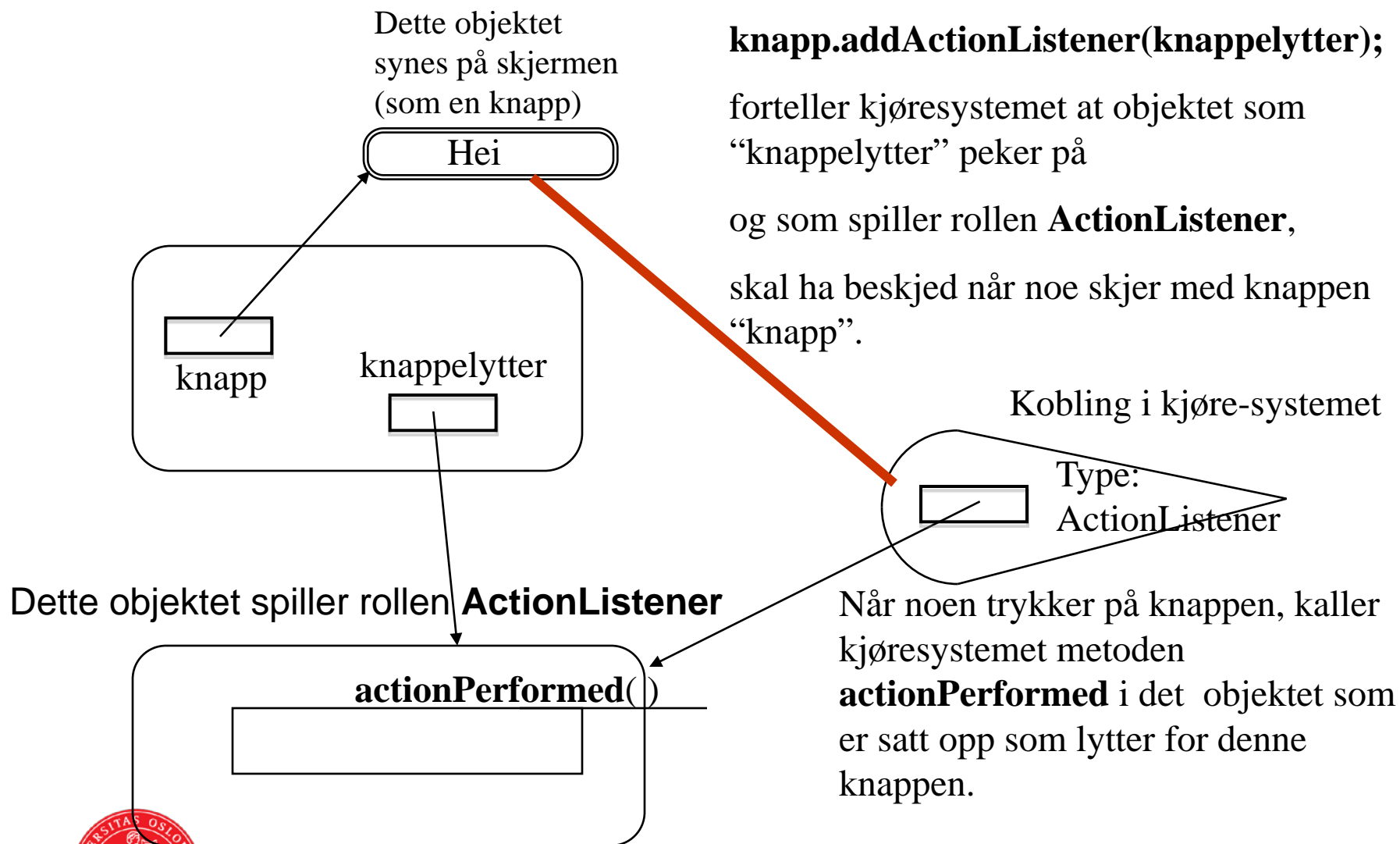
Legger knappen inn i vinduet

```
 knapp.addActionListener(knappelytting);
```

NYTT:
Sier fra hvem som skal lytte
etter trykk på denne
knappen
(mer på neste side)

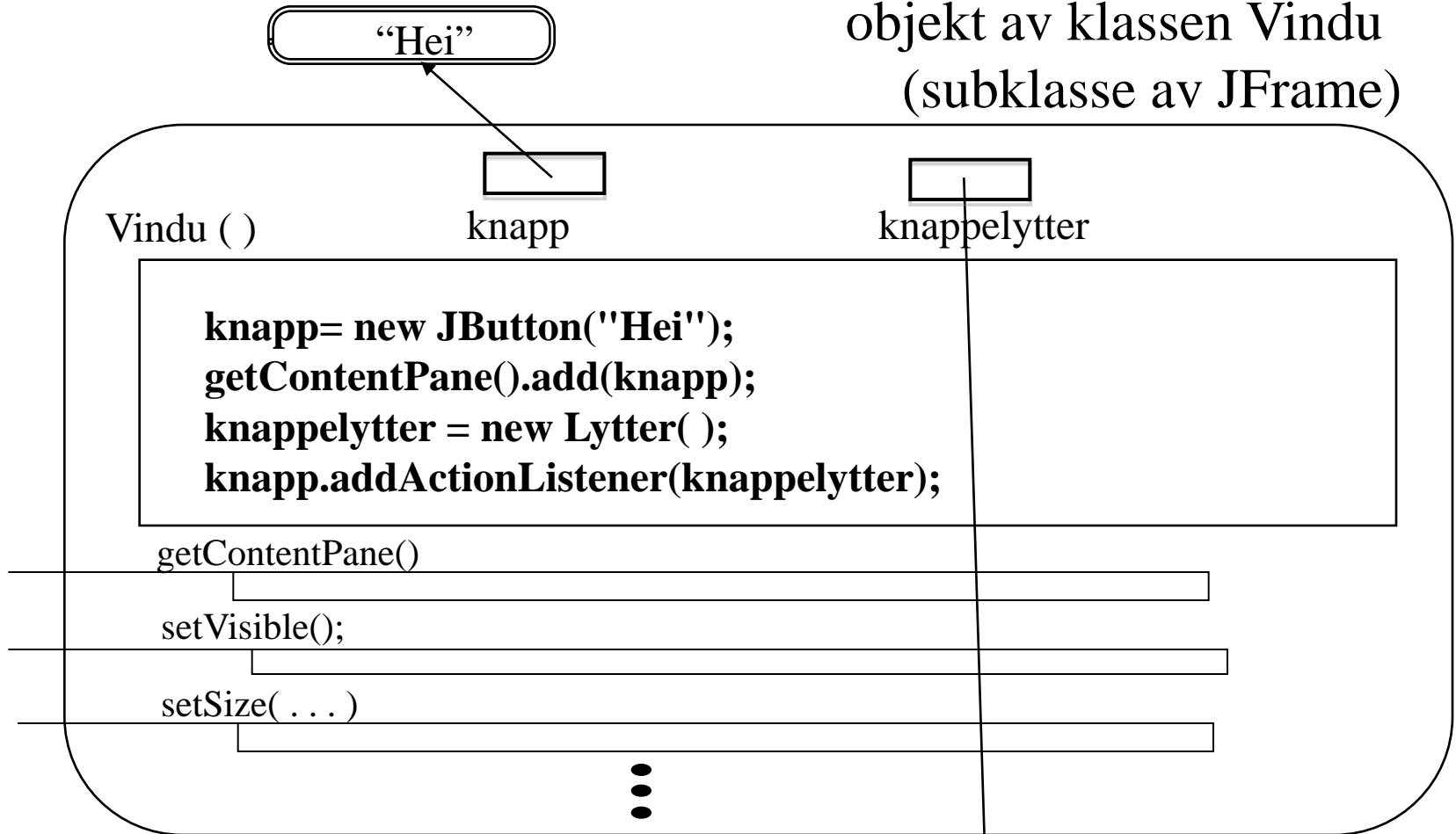


Hva skjer ved et knappetrykk?



objekt av den ferdiglagde
klassen JButton

objekt av klassen Vindu
(subklasse av JFrame)



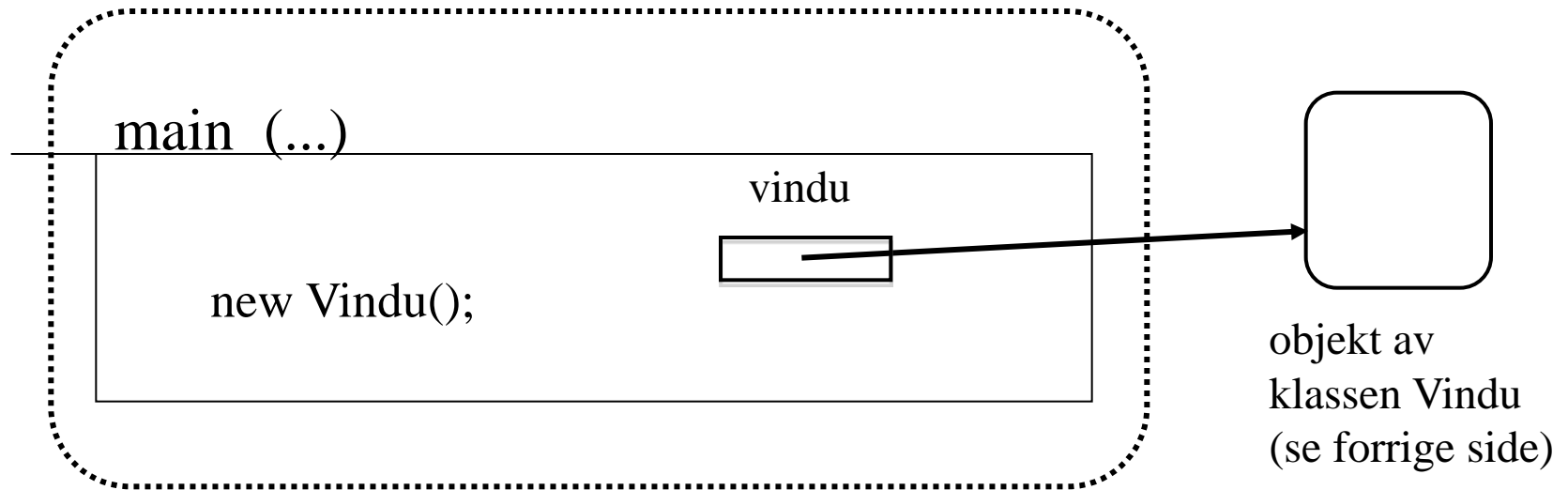
objekt av klassen Lytter (implements **ActionListener**)

actionPerformed (...)

System.out.println("Noen sa hei til meg");



Klassedatastrukturen til class Vindu



Vindu-objektet (konstruktøren)
ordner så resten selv.

Fullstending mini-program med bare en knapp (og utskrift i DOS-vinduet)

```
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;
```

```
public class Vindu extends JFrame  
{ private JButton knapp;  
  private Lytter knappelytter;
```

```
  public Vindu( ) {  
    super("Hei test");  
    Container samling = getContentPane();  
    samling.setLayout(new FlowLayout());  
    setSize(300,200);  
    knapp= new JButton("Hei");  
    samling.add(knapp);  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    setVisible(true);  
    knappelytter = new Lytter( );  
    knapp.addActionListener(knappelytter);  
  } // slutt Vindu konstruktør
```

Metoden
getContentPane()
i JFrame returnerer
"bildeflaten"
til dette vinduet

Her lager
vi et
lytterobjekt

Her kobles knappen
opp mot dette
lytterobjektet



Program forts.

```
public static void main(String[] args) {  
    JFrame vindu = new Vindu();  
} //slutt main
```

```
class Lytter implements ActionListener {  
    public void actionPerformed(ActionEvent e)  
        { System.out.println("Noen sa hei til meg"); }  
} // slutt class Lytter
```

```
} // slutt class Vindu
```

Dette er
lytterklassen
som spiller
rollen
ActionListener

Kjøring (Windows)

```
C:\javaprogram> javac Vindu.java
```

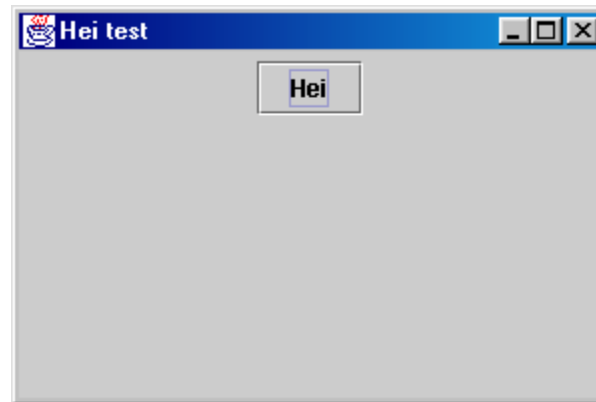
```
C:\javaprogram> java Vindu
```

```
Noen sa hei til meg
```

```
Noen sa hei til meg
```

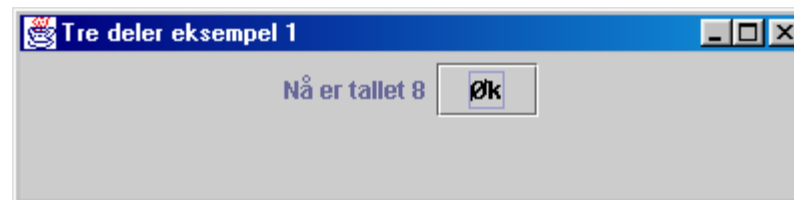
```
Noen sa hei til meg
```

```
C:\javaprogram>
```



Nytt, viktig og enkelt eksempel: Mini-program med tre deler

- Kontroll
 - Sørger for at datastrukturen blir manipulert på riktig måte (økt med én hver gang knappen trykkes)
 - Sørger for at forandringer i datastrukturen blir skrevet ut.
- Utsyn
 - En knapp som gir beskjed til kontrollen hver gang den blir trykket på
 - Kan skrive ut en tekst som angir hvordan datastrukturen nå ser ut (hvor stort tallet er blitt)
- Modell
 - Datastrukturen er bare ett tall med to operasjoner (legg til én og les av)
- Model-View-Controller (MVC) - et mønster for programmering



Objekt av class Kontroll

```
Kontroll(){
    dataStrukt = new Modell(4);
    vindu = new Utsyn(this); }
```

knappTrykket()

```
int tall;   dataStrukt.oppdater();
tall= dataStrukt.hentNyVerdi();
vindu.skrivUt(tall);
```

dataStrukt

vindu

int antall;

antall

Modell (int tall)

```
{ antall = tall; }
```

int hentNyVerdi()

```
{ return antall; }
```

oppdater()

```
{ antall ++; }
```

Objekt av class JLabel

Tallet er 17

tekst

Objekt av class JButton

Øk

knapp

kntrl

Utsyn (Kontroll kntrl)

```
super("Tre deler eksempel 1");
```

```
kntrl = kntrl;
```

```
tekst = new JLabel("Her kommer .. ");
getContentPane(). add(tekst);
```

```
knapp= new JButton("Øk");
getContentPane(). add(knapp);
```

```
knapp.addActionListener(new KnappLytter);
```

skrivUt(int tall)

```
{ tekst.setText("Tallet er " + tall); }
```

Objekt av class Modell

Objekt av class Utsyn



```
public class Kontroll {
```

```
    Modell dataStrukt;
```

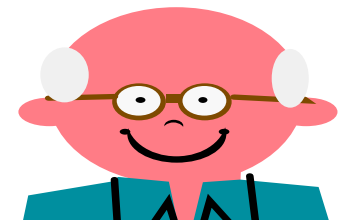
```
    Utsyn vindu;
```

```
    Kontroll(){  
        dataStrukt = new Modell(4);  
        vindu = new Utsyn(this);  
    }
```

```
    public static void main(String[] args) {  
        new Kontroll();  
    } //end main
```

```
    public void knappTrykket(){  
        int tall;  
        dataStrukt.oppdater();  
        tall= dataStrukt.hentNyVerdi();  
        vindu.skrivUt(tall);  
    }  
} // slutt Kontroll
```

- **main er minimal**
- **Kontroll-konstruktøren lager de to andre objektene**
- **actionPerformed (neste side) kaller knappTrykket**



```
class Utsyn extends JFrame {
    JButton knapp; JLabel tekst;
    Kontroll kntrl;
```

Peker tilbake til Kontroll-objektet

```
    class KnappeLytter implements ActionListener { // indre klasse
        public void actionPerformed(ActionEvent e) {
            kntrl.knappTrykket();
        }
    }
```

...og her brukes den

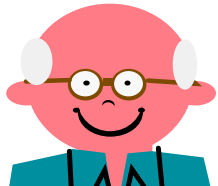
```
    public Utsyn(Kontroll kontrl){
        super("Tre deler eksempel 1");
        setFont(new Font("Serif", Font.PLAIN,18));
        setSize(400,100);
        kntrl = kontrl;
        getContentPane().setLayout(new FlowLayout());
        tekst = new Label("Her kommer en melding");
        getContentPane().add(tekst);
        knapp= new Button("Øk");
        getContentPane().add(knapp);
        knapp.addActionListener(new KnappeLytter());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    } // slutt Utsyn konstruktør
```

```
        public void skrivUt(int tall){ tekst.setText(" Tallet er nå: " + tall + " ");}
    } // slutt class Utsyn
```



Datastruktur

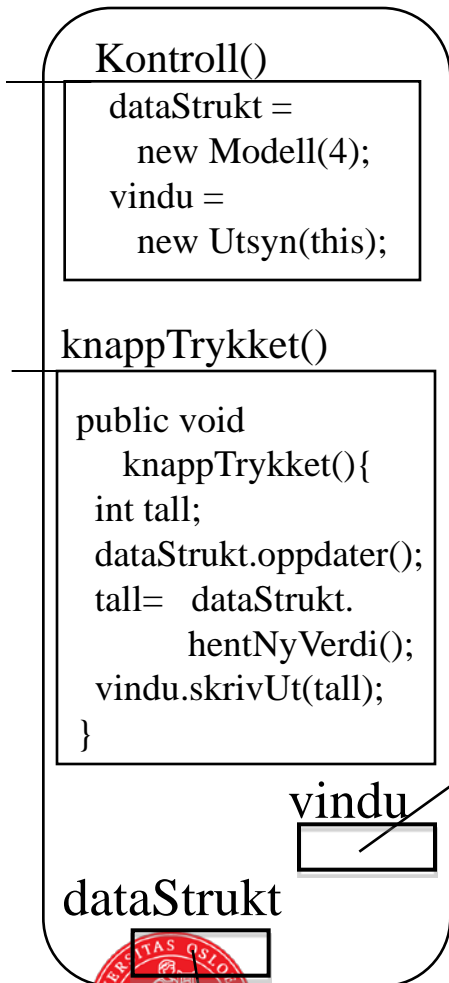
```
class Modell{  
  
    private int antall;  
  
    Modell (int tall) { antall = tall;}  
  
    public int hentNyVerdi( ) { return antall; }  
  
    public void oppdater( ) { antall ++;}  
  
} // slutt klass Modell;
```



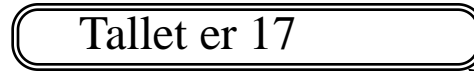
**Som sagt:
Enkleste datastruktur vi kan tenke oss**



Objekt av class Kontroll



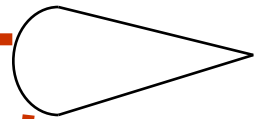
Objekt av class JLabel



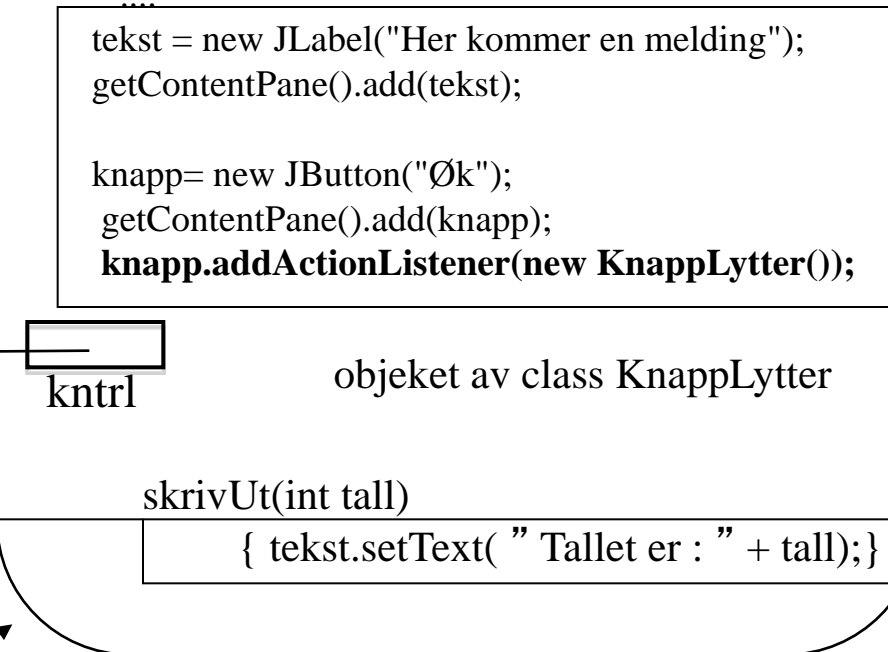
Objekt av class JButton



Kjøresystemet



Utsyn (Kontroll kontroll)



Objekt av class Utsyn

Objekt av den indre class KnappLytter (i Utsyn-objektet over)

actionPerformed (ActionEvent e)

```

{
  kntrl.knappTrykket();
}
  
```



Flere knapper: Flere lytterklasser:

```
enKnapp = new JButton("Noe");  
enKnapp.addActionListener(new NoeSkjer());
```

```
stoppKnapp = new JButton("Stopp");  
stoppKnapp.addActionListener(new Stopp());
```

```
class Stopp implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        setVisible(false); // omliggende vindu  
        ...    }  
}
```

```
class NoeSkjer implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        ....    }  
}
```

Det er mulig å bruke
en klasse med en
ActionListener-metode,
og test på hvilken knapp
som ble trykket på
(neste side)



Alternativt: Ett lytte-objekt

```
class Forskjellig implements ActionListener {  
  
    public void actionPerformed(ActionEvent e) {  
        if (e.getSource() == stoppKnapp) {  
            setVisible(false);  
            ...  
        } else if (e.getSource() == enKnapp) {  
            ...  
        }  
    }  
}
```

```
ActionListener lytter = new Forskjellig();
```

```
enKnapp = new JButton("Noe");  
enKnapp.addActionListener(lytter);
```

```
stoppKnapp = new JButton("Stopp");  
stoppKnapp.addActionListener(lytter);
```

