INF1010

Grafisk brukergrensesnitt med Swing og awt del 2

GUI (Graphical User Interface)-programmering

- Tidligere
  - Hvordan få laget et vindu på skjermen
  - Hvordan legge ulike komponenter i vinduet (trykknapper, tekstfelter, tekst, bilder,)
  - Enkel behandling av knappetrykk mm.
  - Layout av vinduer

- I dag:
  - Repetisjon om å lytte på knappetrykk med musa
  - Grafikk (tegning i vinduet)
  - Mer om input fra brukeren via vinduer
  - Eventmodellen (fange opp hendelser) og Javas store bibliotek for det
    - Hendelser (events)
    - Lyttere med grensesnitt eller ”Adaptere”
    - Lyttemetodene

Vi lærte sist å lage vinduer

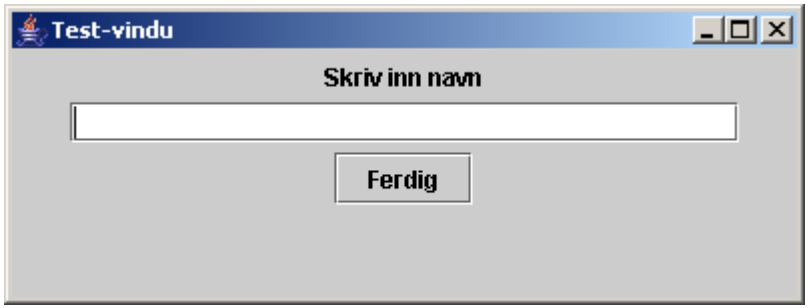
JFrame er selve bildet (rammen)

kompo-nenter (inne i en Container)

Elementene kan legges rett i ramma (JFrame) eller i en beholder av type Container som vi får tak i ved å kalle getContentPane( )

Om å lage et vindu

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

public class Vindu extends JFrame {
    JLabel etikett;       JTextField tekstfelt;       JButton knapp;
    public Vindu( ) {
      setTitle(”Test-vindu");
      Container samling = getContentPane();
      samling.setLayout(new FlowLayout());
      etikett = new JLabel("Skriv inn navn");
      samling. add(etikett);
      tekstfelt = new JTextField(30);
      samling. add(tekstfelt);
       knapp = new JButton(”Ferdig");
      samling. add(knapp);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setBounds(100, 250, 400,150);
      setVisible(true);
    }
}
```



Mangler å lytte på knappen/tekstfeltet

Samme eksempel, men legger elementene rett i vinduet (JFrame)

```
// fjernet import-setninger og klassen med main-metoden

class Vindu extends JFrame {

  JLabel etikett;
  JTextField tekstfelt;
  JButton knapp;

  Vindu( ) {
     setTitle("Test-vindu");
     // Container samling = getContentPane();
     setLayout(new FlowLayout());
     etikett = new JLabel("Skriv inn navn");
     add(etikett);
     tekstfelt = new JTextField(30);
     add(tekstfelt);
     knapp = new JButton("Ferdig");
     add(knapp);
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     setBounds(100, 250, 400,150);
     setVisible(true);
  }
}
```

De viktigste byggesteinene i GUI-programmering

Vinduer/rammer/lerreter

JFrame
JPanel

setLayout()
add()
paintComponent()

Utlegg/layouts

BorderLayout
FlowLayout
GridLayout
…..

Elementer/objekter (subklasser av JComponent)

JLabel, JTextField, JButton, …
addActionListener()

Lyttergrensesnitt (alternativt adaptere)

KeyListener, ActionListener, ItemListener, ….

actionPerformed(**ActionEvent e**), m.fl.....

For å lytte på en komponent (repetisjon)

1. **Lag en metode som beskriver hva som skal gjøres når en bestemte hendelser inntreffer. Eksempel:**

   public void actionPerformed(ActionEvent e) { .... }

Metodene skal kalles av Java og må ha et helt bestemte navn som er forhåndsbestemt av Java-biblioteket (polymorf metode).

2. Put metoden inn i en lytterklasse (f.eks. **class MinLytter**) som implementerer et bestemt interface (også fra Java-biblioteket).

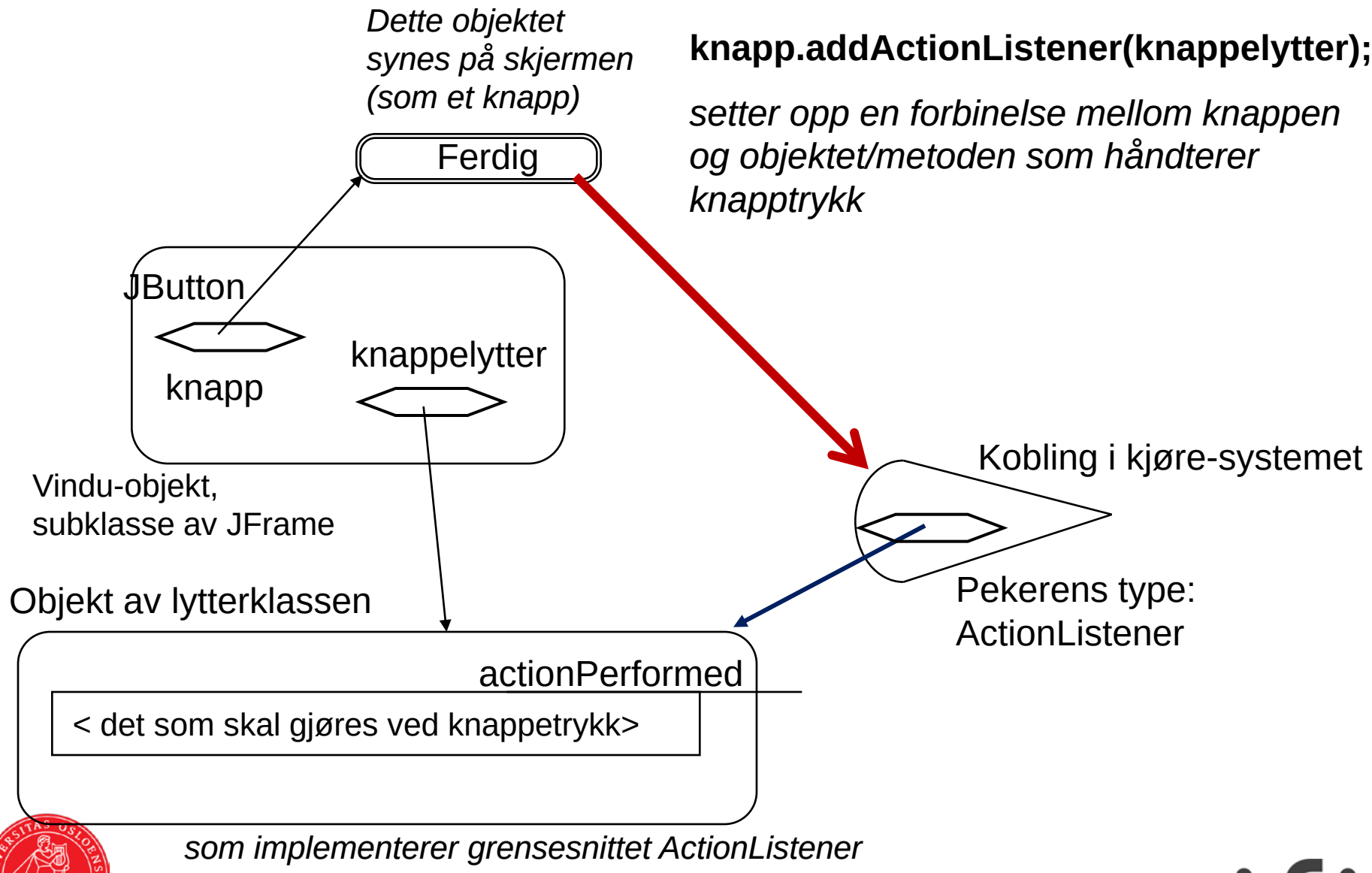
3. **Lag et objekt av lytterklassen og send referansen til dette objektet til den komponenten (f.eks. knappen) som dette objektet skal lytte på. Eksempel:**

   knappen.addActionListener(new MinLytter());

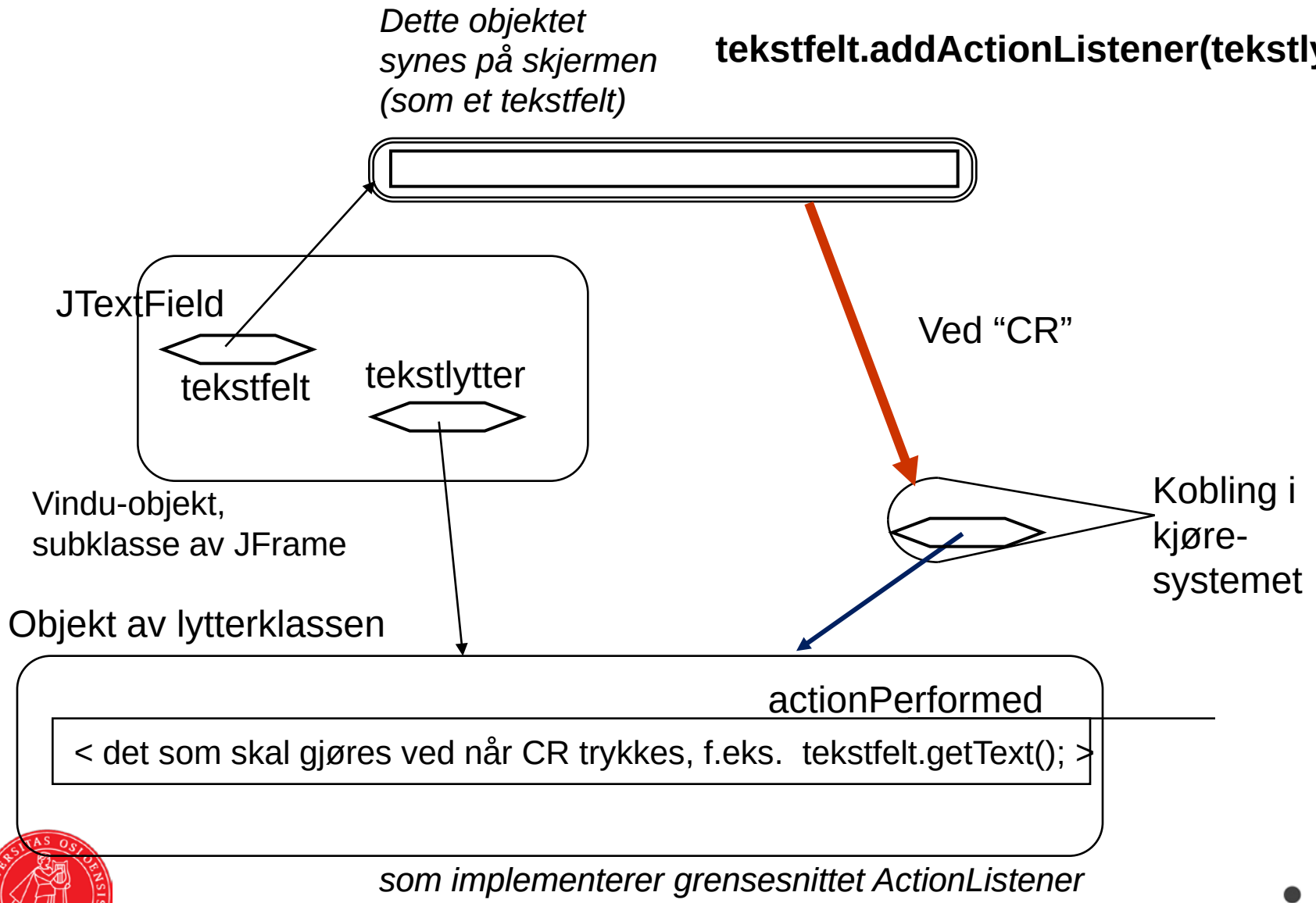
   ↑

   nesten et frittsvevende objekt
   (bare kontakt fra Runtime-systemet)

Vi har lærte å lytte på knapper



actionPerformed kan også lytte på "CR" i tekstfelt

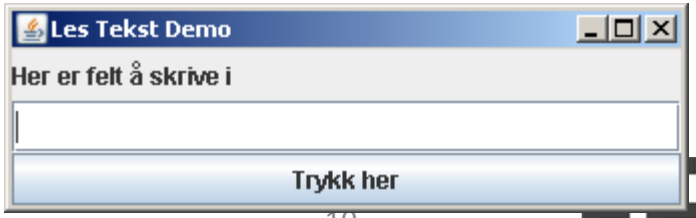


Fullstending mini-program

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

public class Vindu extends JFrame  {
                JLabel ledetekst =       new JLabel("Her er felt å skrive i");
               JTextField tekstfelt =  new JTextField(30);
               JButton knapp =         new JButton("Trykk her");

 public Vindu( )  {
               super("Les Tekst  Demo");
}
 void lagVindu( ) {
               tekstfelt.setEditable(true);
               Container lerret =  getContentPane();
               lerret.setLayout(new GridLayout(3, 1));
               lerret.add(etikett);
               lerret.add(tekstfelt);
               lerret.add(knapp);
               MinLytter l1 = new MinLytter();
               tekstfelt.addActionListener(l1);
               knapp.addActionListener(l1);
               setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
               pack();
               setVisible(true);
}

public static void  main(String[] argumenter) {
               new Vindu().lagVindu();
}
```

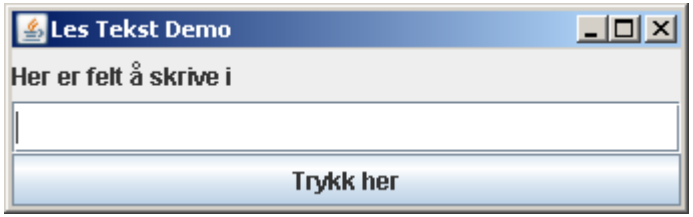
Her lager vi et lytterobjekt

Her kobles både knappen og tekstfeltet opp mot dette lytterobjektet



Lytterklassen

```
class Lytter  implements   ActionListener {
        public void actionPerformed(ActionEvent e)  {
                        String st = tekstfelt.getText();
                        tekstfelt.setText("");
                        System.out.println(st);
         }
  } // slutt class Lytter

} // slutt class Vindu
```



Flere knapper: Flere lytterklasser:

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

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

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

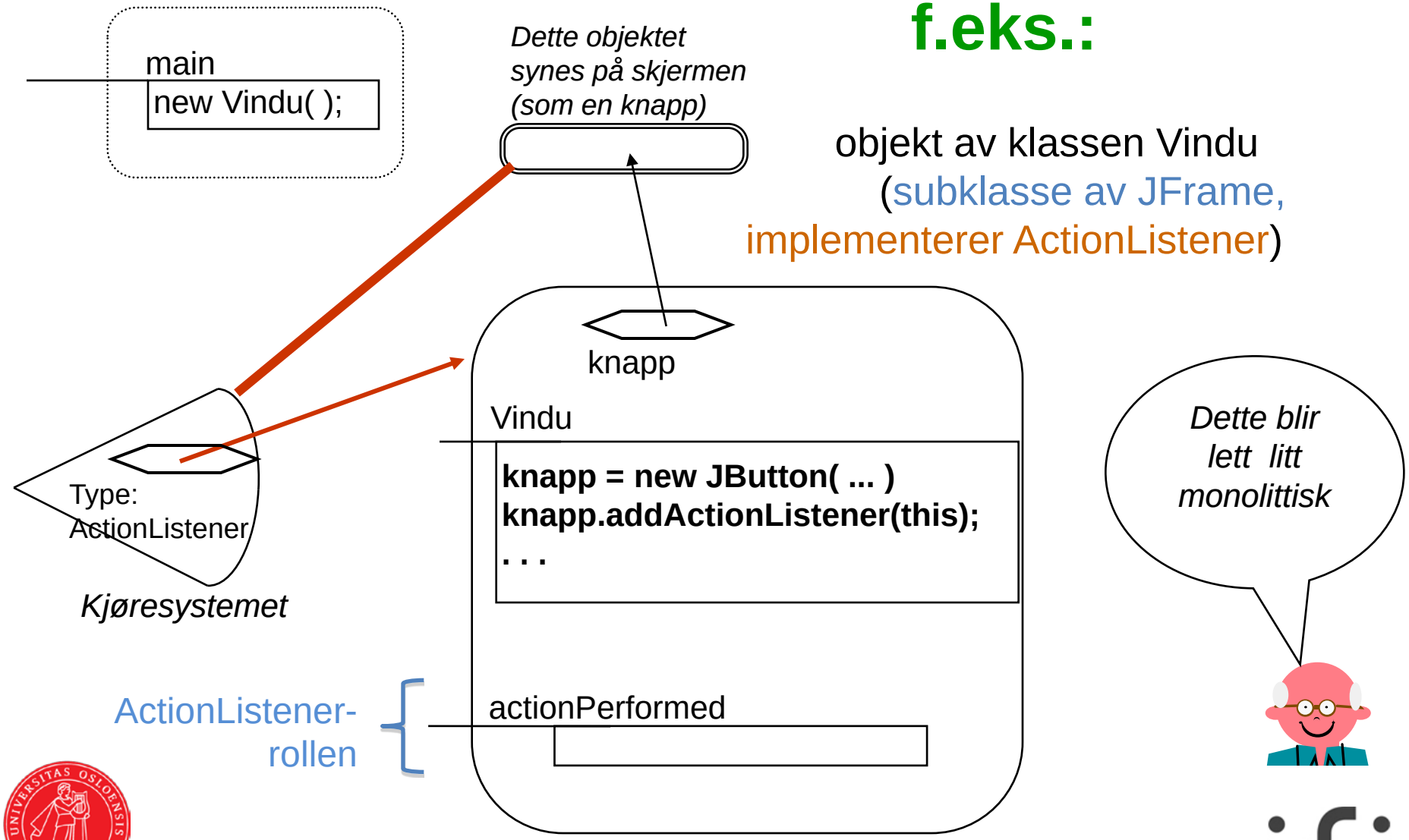
```
class  Stopp implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                setVisible(false); // omliggende vindu
            . . . . .
        }
}

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

Flere knapper: en lytte-metode

```
class  Forskjellig  implements ActionListener {

        public void actionPerformed(ActionEvent e) {

                    if (e.getSource( ) = = stoppKnapp) {
                            setVisible(false);
                                . . .
                     }   else if (e.getSource( ) = = enKnapp) {
                                . . .
                     }
          }
}
```

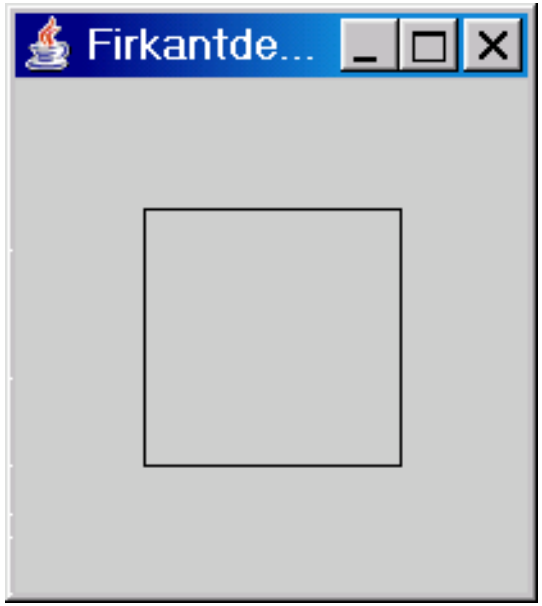
```
ActionListener lytter = new Forskjellig();

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

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

Det er også mulig å slå sammen lytter-objektet med andre objekter f.eks.:



2D grafikk – tegning i (en del av) vinduet

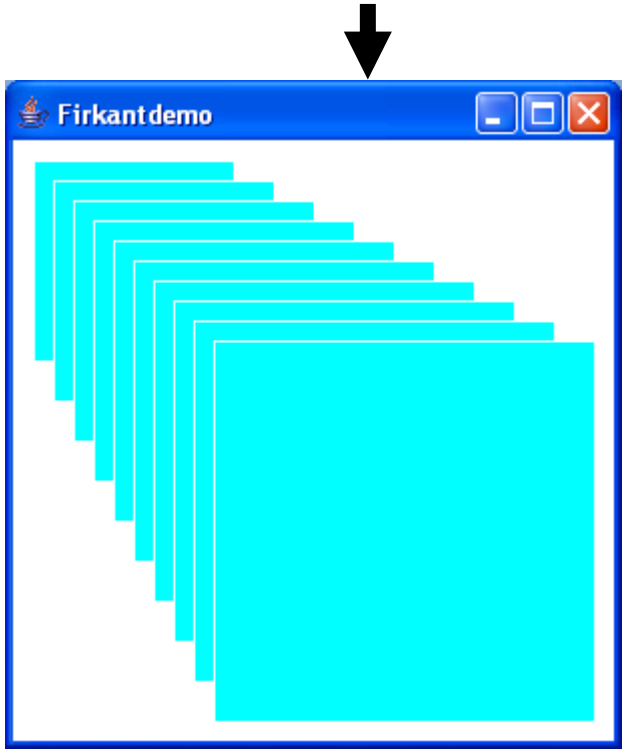
- Alle klassene i Swing og awt har paint() metoder som kalles når systemet får beskjed om å vise noe på skjermen – eks. setVisble(true).
- Vi kaller aldri disse direkte – bare indirekte via repaint() og setVisible(true) metodene
- Vi lager selv vår versjon av paint() eller paintComponent() når vi vil lage grafikk i en flate – f.eks. et JPanel
- Når metodene paint() eller paintComponent() som vi har laget, kalles fra systemet, får de med en parameter, Graphics g, og det er på denne 'g' vi skal tegne



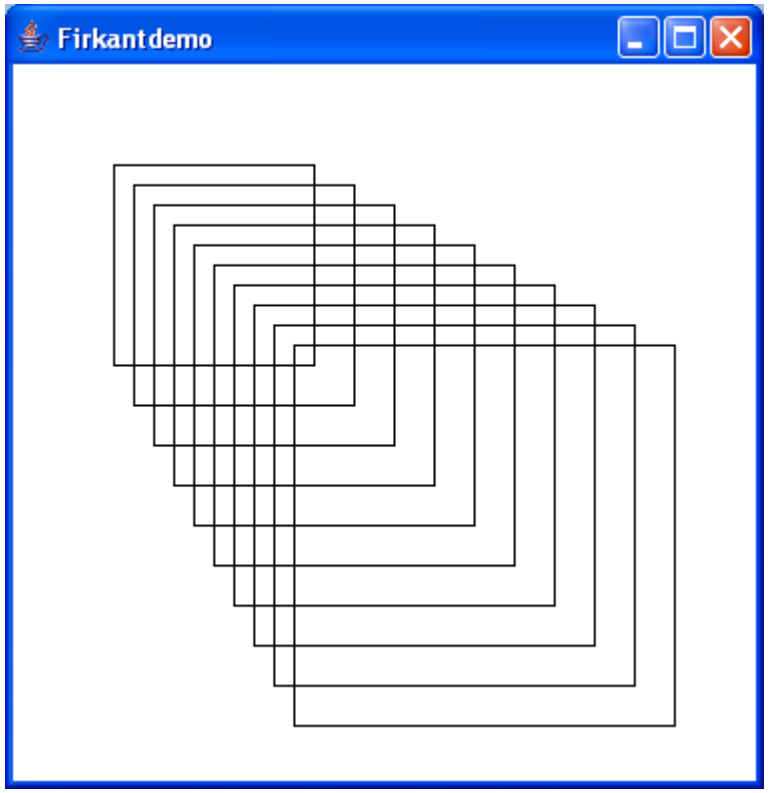
```
import javax.swing.*;
import java.awt.*;

class Firkant extends JPanel {
        Firkant() {
                // angir foretrukket størrelse på dette lerretet.
                setPreferredSize(new Dimension(200, 200));
        }
        public void paintComponent(Graphics g) {
                // Her tegner vi
                g.drawRect(50, 50, 100, 100);
        }
}

class FirkantDemo extends JFrame {
        FirkantDemo() {
                setTitle("Firkantdemo");              // enda en måte å sette tittel på
                Container lerret = getContentPane();   // peker til vindusflaten
                JPanel panel = new Firkant();          // lag panel (med ”firkant”)
                lerret.add(panel);                     // legg firkanten inn
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                pack( );                               //passe stort vindu
                setVisible(true);                      // gjør alt synlig
        }
        public static void main(String[] args) {
                new FirkantDemo();
        }
}
```

10 firkanter

```
public void paintComponent (Graphics g) {
    for (int i = 0; i < 100; i += 10){
            g.setColor(Color.cyan);
            g.fillRect(10+i, 10+i, 100+i, 100+i);
             g.setColor(Color.white);
            g.drawRect(10+i, 10+i, 100+i, 100+i);
}
```




```
public void paintComponent (Graphics g) {
        for (int i = 0; i < 100; i += 10)
                g.drawRect(50+i, 50+i, 100+i, 100+i);
}
```



Tekst og linjer

Bytter ut class Firkant i eksempelet:

```
class Firkant extends JPanel {
int topp = 50,  kant= 100, vside= 40;
    Firkant( ) { setPreferredSize(new Dimension(vside+kant+vside, topp+kant+30));  }

     public void paintComponent(Graphics g) {
               super.paintComponenet(g);   // prøv med og uten
              g.setColor(Color.cyan);
                        g.fillRect(vside, topp , kant, kant);
              g.setColor(Color.black);
               Font skrift = new Font("SansSerif",Font.BOLD,12);
               g.setFont(skrift);
               g.drawString("Firkant",vside+5,topp + kant/2);
               g.setColor(Color.white);
               g.drawLine(vside+5,topp+kant/2+5,vside+kant-5,topp+kant/2+5);
    }
}
```

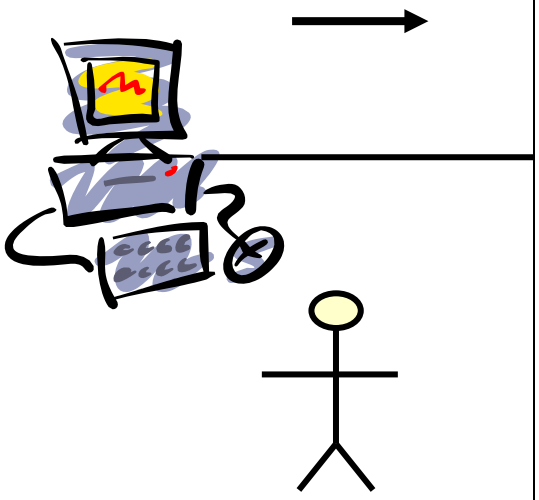
Vise fram et bilde

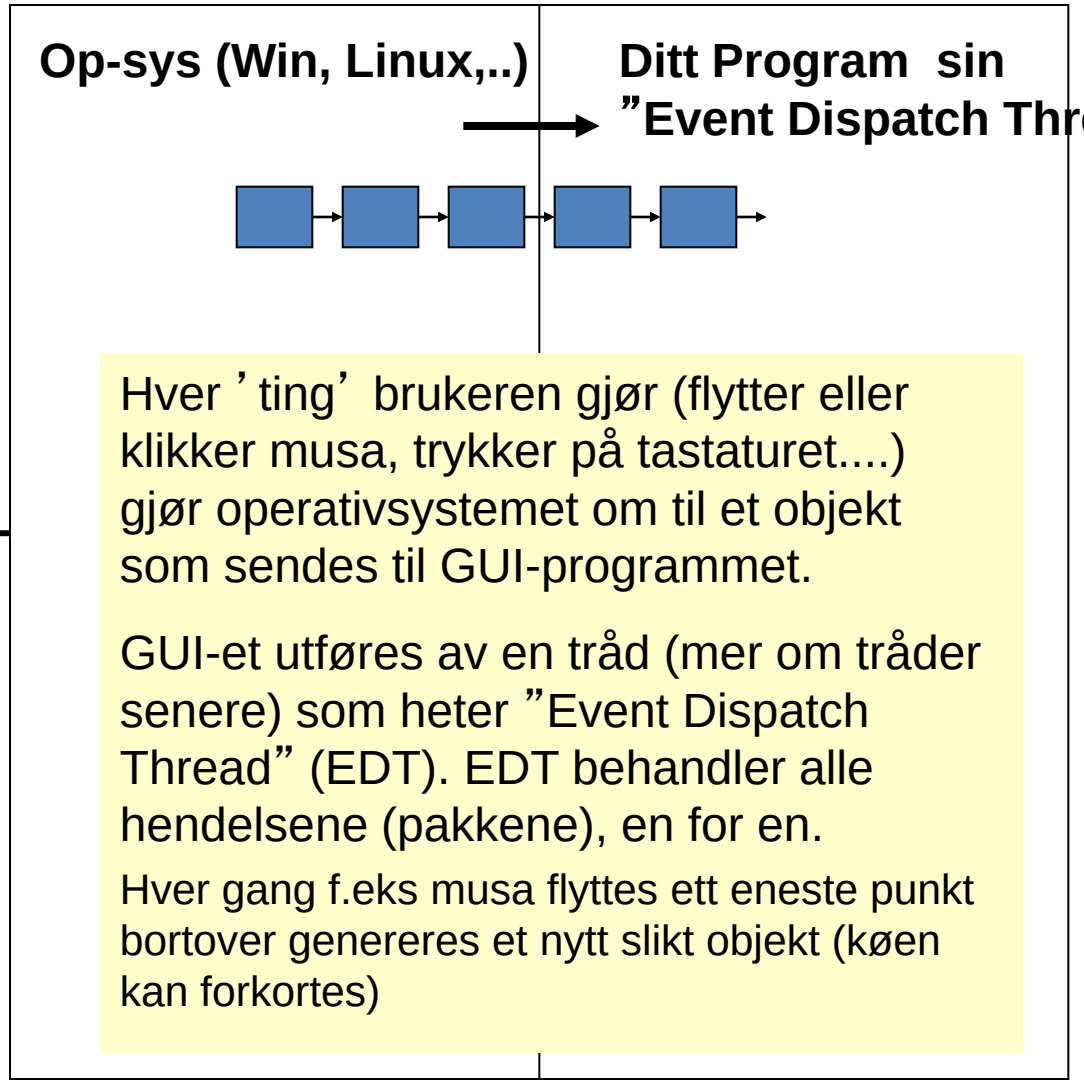
```
class BildeVindu extends JFrame {
    BildeVindu () {
        setTitle("BildeDemo");
        setLayout(new FlowLayout());
        BildeLerret bpanel = new BildeLerret
              (new ImageIcon("bilde.jpg").getImage());
        add(bpanel);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack(); setVisible(true);
    }
}

class BildeLerret extends JPanel {
     private Image bilde;

     public BildeLerret(Image bilde) {
      this.bilde = bilde;
      Dimension storleik =
         new Dimension(bilde.getWidth(null),
                       bilde.getHeight(null));
      setPreferredSize(storleik);
      setMinimumSize(storleik);
      setMaximumSize(storleik);
      setSize(storleik);
      setLayout(null);
     }

     public void paintComponent(Graphics g) {
      g.drawImage(bilde, 0, 0, null);
     }

}
```

Hva kan en bruker gjøre – typer av hendelser

- Bevege musa – innenfor en av våre vinduer
  - inn i vinduet
  - ut av vinduet
  - over en av komponentene (trykk-knapp, tekstfelt,..) - hvilken
- Klikke musa – hvilken knapp eller hvor (x,y i vinduet)
  - klikke ned
  - slippe opp
- Taste en tast – hvilken
  - trykke ned
  - slippe opp
- Trykke ”CR” i et tekstfelt
- . . . .

GUI og hendelser





Hver ’ting’ brukeren gjør (flytter eller klikker musa, trykker på tastaturet....) gjør operativsystemet om til et objekt som sendes til GUI-programmet.

GUI-et utføres av en tråd (mer om tråder senere) som heter ”Event Dispatch Thread” (EDT). EDT behandler alle hendelsene (pakkene), en for en.

Hver gang f.eks musa flyttes ett eneste punkt bortover genereres et nytt slikt objekt (køen kan forkortes)

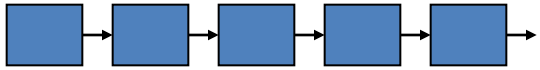
Mer om parameteren (subklasse av EventObject)

for eksempel: public void actionPerformed(**ActionEvent e**)

- Inneholder ”få” ting
  Størrelsen av objektet som **e** peker på er avhengig av type hendelse :
  - Type hendelse :
    - musebevegelse
    - museklikk,
    - trykk på tast (ned eller opp)
    - ...
  - Verdier, f.eks
    - Museposisjon (x,y) i vinduet
    - Hvilken museknapp trykket
    - Hvilken tast trykket
  - Hvilket vinduskomponent på skjermen dette tilhører
    - Peker til komponent eller tekst
    - Hvordan vet kjøresystemet dette ?

Mer om hendelsesmodellen – en kø per program

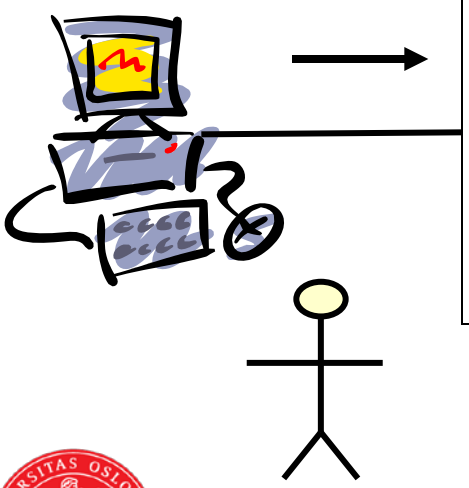
Operativsystem (Win, Linux,..) → **Ditt Program sin "Event Dispatch Thread"**



Operativsystemet har vinduer fra flere programmer på skjermen, men vet hvilket vindu (og da program) en hendelse tilhører.

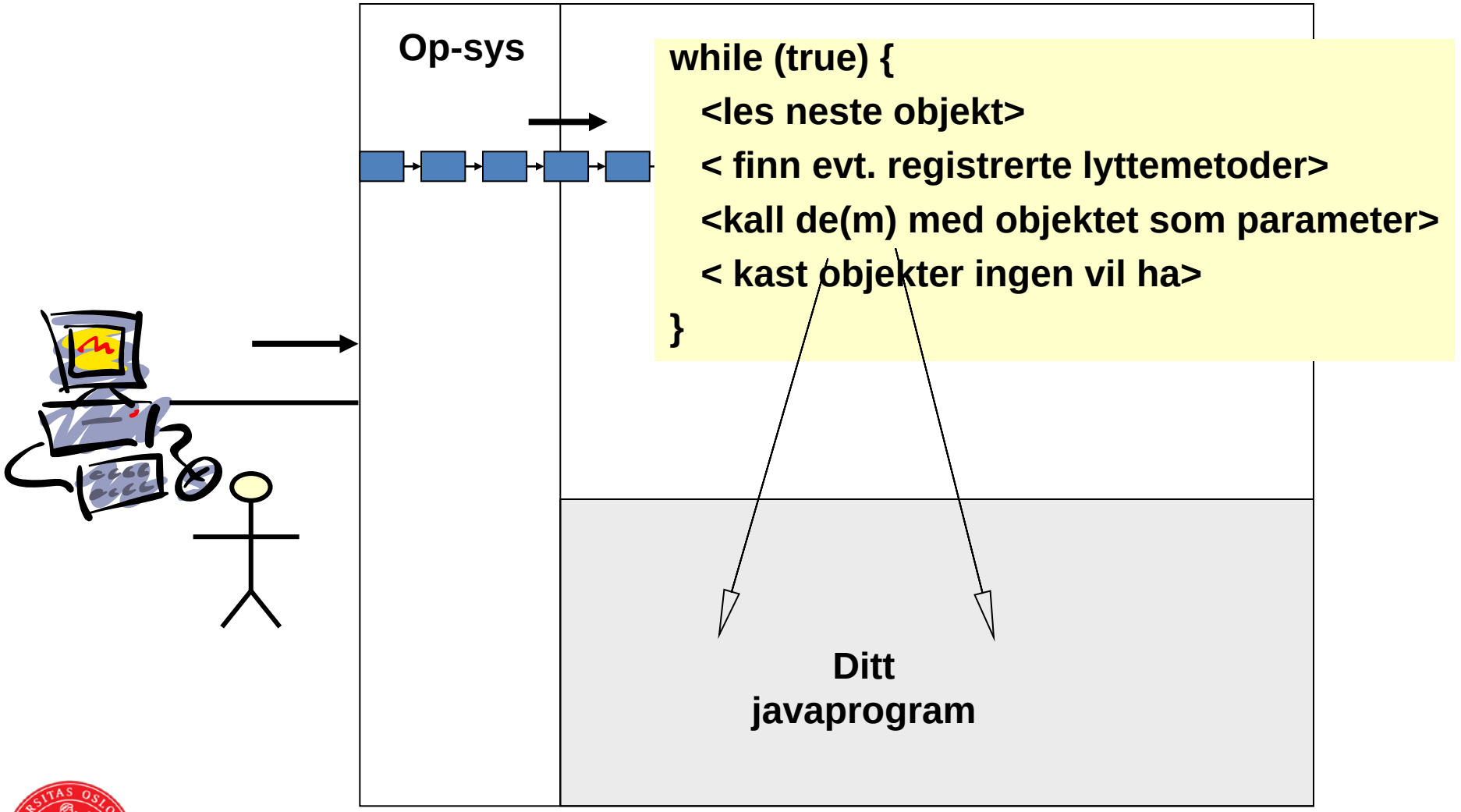
Lager ***en slik kø for hvert program*** som har vinduer på skjermen

"Event Dispatch Thread" (EDT) i hvert program får da bare sine hendelser.



Hvis flere metoder blitt registrert som lyttere av samme type hendelse, får de hver sin kopi av objektet.
Kaster de objektene ingen vil ha

Java fordeler pakkene



Viktigste lyttergrensesnitt

- Grensesnitt og typen av hendelser
  - MouseListener
    - museklikk
  - MouseMotionListener
    - musebevegelser
  - ActionListener
    - Trykk på trykknapper / CR i tekstfelt
  - KeyListener
    - Tastetrykk
  - ItemListener
    - Valg i rullegardin
  - + ca 80 andre lytter-grensesnitt – f.eks:

ListSelectionListener, MenuDragMouseListener, MenuKeyListener, MenuListener, MetaEventListener, MouseInputListener, MouseWheelListener, NamespaceChangeListener, NamingListener, NodeChangeListener, ObjectChangeListener, PopupMenuListener, PreferenceChangeListener, PropertyChangeListener, RowSetListener, SSLSessionBindingListener, TableColumnModelListener, TableModelListener, TextListener, WindowFocusListener, WindowListener, WindowStateListener

Eksempel: Rullegardiner - JComboBox

```
JComboBox styring = new JComboBox ();
styring.addItem("Opp");
styring.addItem("Ned");
styring.addItem("Venstre");
styring.addItem("Høyre");
styring.setEditable(false);
styring.addItemListener(new RulleLytter);
...
```

```
Class RulleLytter implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
        String s = (String) e.getItem();
        if (s.equals("Opp"))      {fart++;} else
        if (s.equals("Ned"))      {fart - - ;} else
        if (s.equals("Høyre"))    {retning = hoyre;} else
        if (s.equals("Venstre"))  {retning = venstre;}
    }
}
```

Hvordan lage et lytteobjekt – to måter:

1 Implementere det riktige **grensesnittet** (KeyListener, ActionListener, ItemListener)

- Husk at da må vi implementere alle metodene i grensesnittet
- Dette gjør vi som oftest bare for noen enkle grensesnitt (der det bare er én metode i hvert grensesnitt) :
  - ActionListener
  - ItemListener

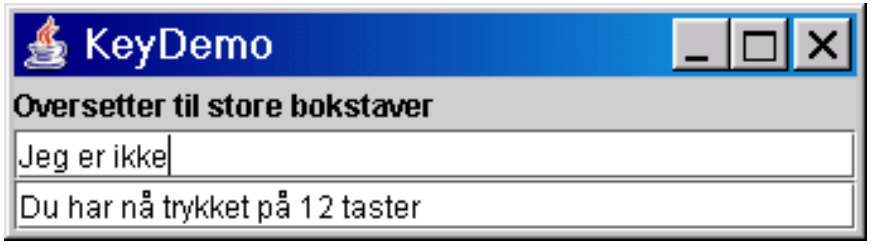
2 Lage en lytterklasse som **subklasse av en ferdiglaget klasse** som allerede er laget og som implementerer lytter-grensesnittet

NY

- De heter alle **Adapter** (KeyAdapter, MouseAdapter,..) og har implementert **tomme metoder** for alle metodene i grensesnittet.
- I en subklasse kan vi da enkelt omdefinere de metodene vi skal bruke (vi slipper å gi kode til resten av metodene)

KeyAdapter - eksempel

- Vil lytte på at brukeren gir CR (= enter-tasten, vognretur på norsk)
- Telle opp hvor mange tastetrykk som tastes inn i øverste felt
- Hvis det er CR, så oversette alt til STORE BOKSTAVER i nederste felt



12 ?

KeyListener

- Metoder i grensenittet:
- char **getKeyChar**()

  Returns the character associated with the key in this event.
- int **getKeyCode**()

  Returns the integer keyCode associated with the key in this event.
- + ca 10 andre

  (så en kan forstå det blir litt tungt å implementere dem alle hvis vi bare trenger en eller to)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;  // Husk – for hendelseshåndtering

class KeyDemo extends JFrame {
        JLabel etikett = new JLabel("Oversetter til store bokstaver");
        JTextField tekstfelt = new JTextField(30),
                svarfelt  = new JTextField(30) ;

        KeyDemo() {
                super("KeyDemo");
                tekstfelt.setEditable(true);
                svarfelt.setEditable(true);
                Container lerret =  getContentPane();
         lerret.setLayout(new GridLayout(3, 1));
                 lerret.add(etikett);
                 lerret.add(tekstfelt);
                 lerret.add(svarfelt);
                tekstfelt.addKeyListener(new Tast());
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                pack();
                setVisible(true);
        }

        class Tast extends KeyAdapter {
                int teller = 0;
                public void keyReleased (KeyEvent e) {
                        if (e.getKeyCode() == KeyEvent.VK_ENTER) {
                                String s = tekstfelt.getText();
                                svarfelt.setText(s.toUpperCase());
                        } else {
                                teller++;
                                svarfelt.setText("Du har nå trykket på " + teller +" taster");
                          }
                  }
        }
```

```
class Tast extends KeyAdapter {
    int teller = 0;
    public void keyReleased (KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_ENTER) {
                            String s = tekstfelt.getText();
                            svarfelt.setText(s.toUpperCase());
                }
            else {
                        teller++;
                            svarfelt.setText("Du har nå trykket på " + teller +" taster");
                }
    }
 }

  public static void main(String[] args) {
                new KeyDemo();
  }
}
```

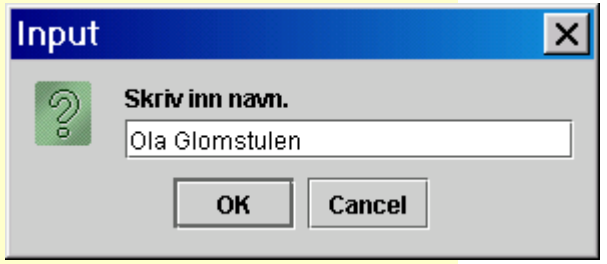
svarfelt.setEditable(false);

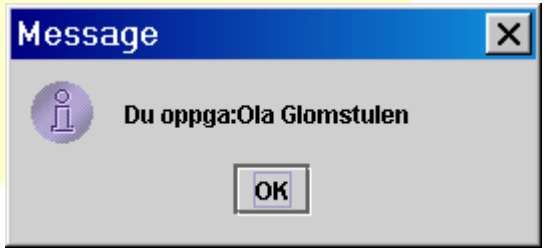
Hvordan er dette mulig?

Ferdigprogrammerte vinduer for én opplysning

- JOptionPane inneholder en rekke ferdige små-vinduer
- De er alle modale (systemet henger til vi har svart)
- Klasser som nytter disse bør være subklasse av JComponent
- Disse finnes i ulike varianter , enkle og mer omfattende parametre
- Problem: Vanskelig å skrive rene norske vinduer

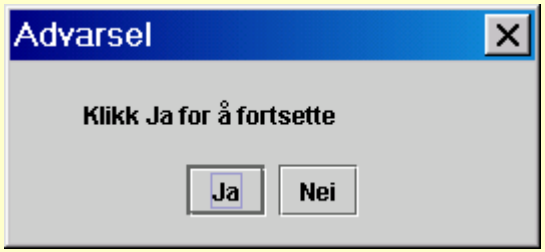
```
import javax.swing.*;
import java.awt.*;

class Dialog1Test extends JComponent{

    public static void main(String[] args) {
        Dialog1Test d = new Dialog1Test();
        String s = JOptionPane.showInputDialog(d, "Skriv inn navn.");

        JOptionPane.showMessageDialog(d, "Du oppga:"+ s);
        }
}
```

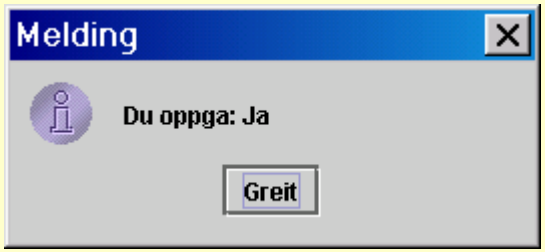




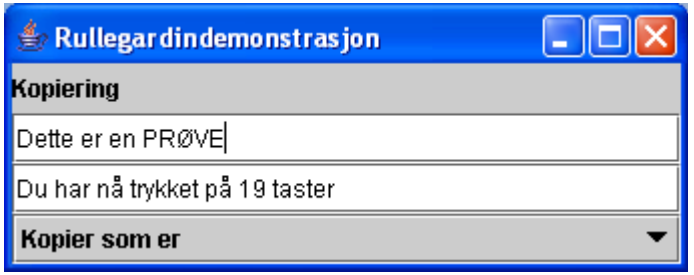
Skal alt på norsk må vi bruke OptionDialog

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

class Dialog2Test {

        public static void main(String[] args) {
        String [] valg = { "Ja", "Nei" };
        int i = JOptionPane.showOptionDialog(null, ”  Klikk Ja for å fortsette",
                ”Advarsel", JOptionPane.DEFAULT_OPTION,
                JOptionPane.PLAIN_MESSAGE, null, valg, valg[0]);

         String [ ] svar = { "Greit"};
        JOptionPane.showOptionDialog(null, "Du oppga: " + valg[i], ”Melding",
                JOptionPane.DEFAULT_OPTION,
                JOptionPane.INFORMATION_MESSAGE, null  , svar, svar[0]);
        System.exit(0);
        }
}
```





Rullegardindemonstrasjon

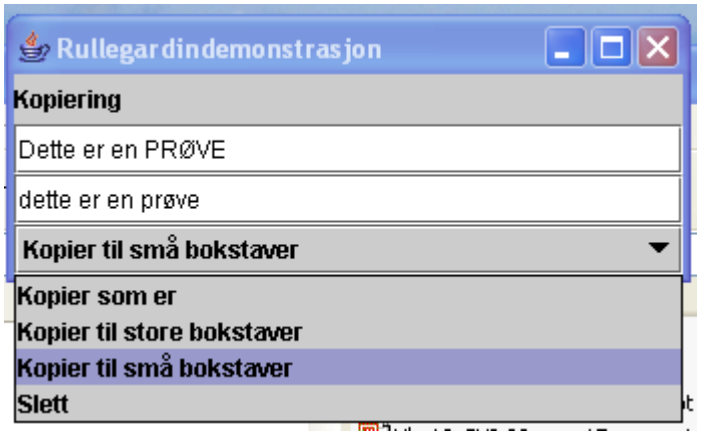



```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;  // Husk for hendelseshåndtering

class RulleDemo extends JFrame {
    JLabel etikett;
    JTextField tekstfelt, svarfelt;
    JComboBox kopi;

    RulleDemo() {
        super("Rullegardindemonstrasjon");
        etikett = new JLabel("Kopiering");
        tekstfelt = new JTextField(30);
         svarfelt  = new JTextField(30) ;
         tekstfelt.setEditable(true);
         kopi= new JComboBox ();
         kopi.addItem("Kopier som er");
         kopi.addItem("Kopier til store bokstaver");
         kopi.addItem("Kopier til små bokstaver");
         kopi.addItem("Slett");
         kopi.setEditable(false);
         Container lerret =  getContentPane();
         lerret.setLayout(new GridLayout(4, 1));
         lerret.add(etikett);
         lerret.add(tekstfelt);
         lerret.add(svarfelt);
         lerret.add(kopi);
```

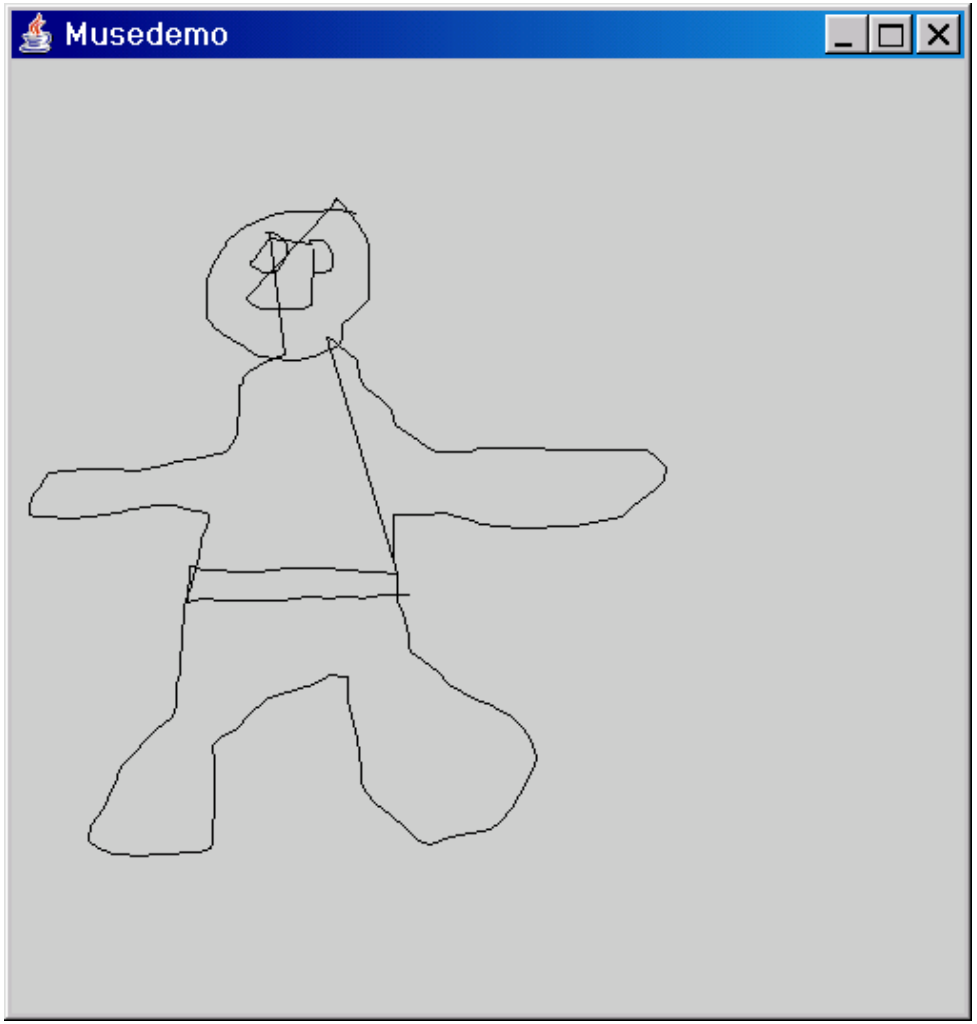

```
         Lytter lytt = new Lytter();
         tekstfelt.addKeyListener(lytt);
         kopi.addItemListener(lytt);
         setDefaultCloseOperation
                              (JFrame.EXIT_ON_CLOSE);
           pack();
           setVisible(true);
    } // slutt konstruktør
```

Rulledemo. forts

```
class Lytter extends KeyAdapter implements  ItemListener {
      int teller =0;
      public void keyReleased (KeyEvent e) {
            teller++;
            svarfelt.setText("Du har nå trykket på " + teller+" taster"):
     }

      public void itemStateChanged(ItemEvent e) {
            String s = (String) e.getItem();
            if (s.equals("Kopier som er"))
                  svarfelt.setText(tekstfelt.getText());
             else if (s.equals("Kopier til store bokstaver"))
                  svarfelt.setText(tekstfelt.getText().toUpperCase());
             else if (s.equals("Kopier til små bokstaver"))
                   svarfelt.setText(tekstfelt.getText().toLowerCase());
            else  if (s.equals("Slett"))
                  { svarfelt.setText(""); tekstfelt.setText(""); }
    }
}

public static void main(String[] args) { new RulleDemo(); }

} // slutt class Rulledemo
```



En Muse-demo – et frihånds tegneprogram



Spesifikasjon

- Vi må lytte på musa (lager en egen klasse) og lagre alle posisjonene musa har vært på (f.eks de siste 10000 posisjonene musa har vært på).
- Hver gang musa røres (knapp-nede-trekkes-til-nytt-sted), legger vi til et nytt punkt og tegner opp hele kurven om igjen
- Samme flaten som tegner, kan også lytte på musa

Oppgave 1: Rett opp en opplagt 'feil' i programmet (sammenhengende strek)

Oppgave 2: Sjekk hva som skjer hvis vi ikke lagrer alle posisjonene musa har vært på (men for eksempel bare de 100 siste)

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


class MuseDemo extends JFrame {
        MuseDemo() {
                setTitle("Musedemo");
                Muse mus = new Muse();
                mus.addMouseMotionListener(mus);
                getContentPane().add(mus, BorderLayout.CENTER);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                pack();
                setVisible(true);
        }
        public static void main(String[] args) {
                new MuseDemo();
        }
}
```

```
class Muse extends JPanel implements MouseMotionListener {
        int [] x = new int[10000],
               y= new int[10000];
        int ant = 0;
        Muse() {    setPreferredSize(new Dimension(500, 500));     }

       // De to metodene i MouseMotionListener:
        public void mouseMoved (MouseEvent e) {   };

        public void mouseDragged (MouseEvent e) {
                if (ant == x.length) ant = 0;
                x[ant]   = e.getX();
                y [ant]  = e.getY();
                ant ++;
                repaint();  // her ber vi om at vår paintComponenet() skal kalles
        }

        public void paintComponent (Graphics g) {
                super.paintComponent(g);
                // Her tegner vi
                for (int i=1; i<ant; i++)
                   g.drawLine(x[i-1], y[i-1], x[i], y[i]);
        }
}
```