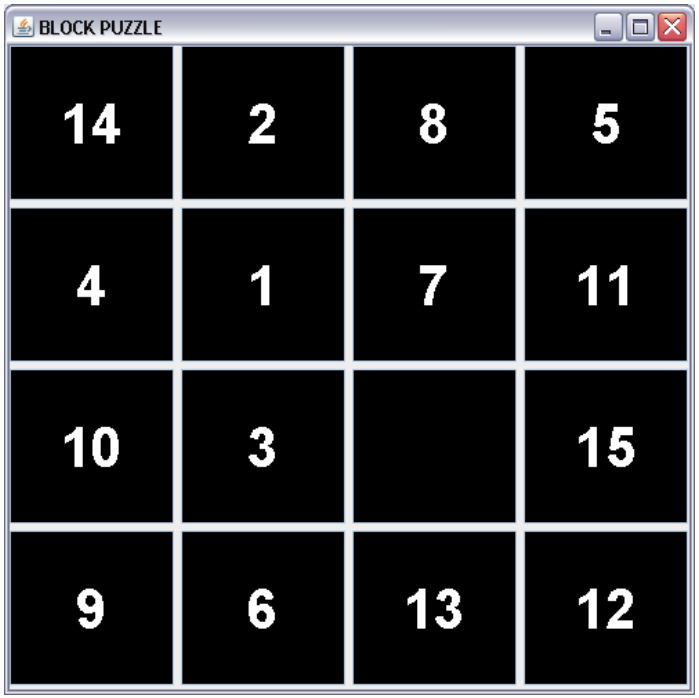


PG2100 – Programmering 2
Forelesning 7

(side 670-682, 846-860)

Agenda

- Innlevering 1 m/ArrayList
- Grafisk input/output med JOptionPane
- Bruke
  - vinduer (JFrame)
  - knapper (JButton)
  - tekstfelt (JTextField)
  - etiketter (JLabel)
- Bestemme layout
- Behandle hendelser

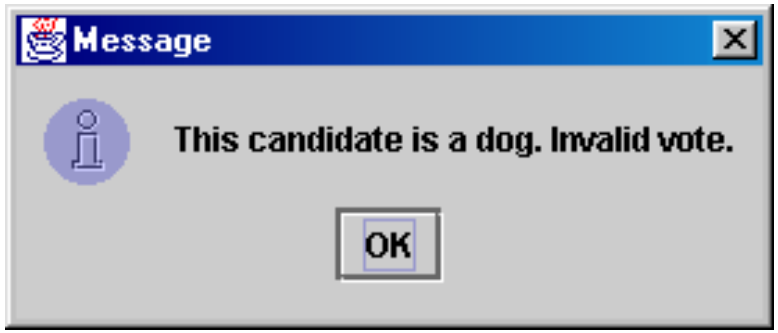
Innleveringen

```java
public class GroceryItem {
    private String name;
    private int quantity;
    private double pricePerUnit;

    public GroceryItem() {

    public GroceryItem(String name, int quantity, double pricePerUnit) {

    public void setName(String name) {

    public void setQuantity(int quantity) {

    public void setPricePerUnit(double pricePerUnit) {

    public String getName() {

    public int getQuantity() {

    public double getPricePerUnit() {

    public double getCost() {

    public String toString() {
}
```

```java
public class GroceryList {
    private GroceryItem[] groceryList;
    private int numberOfItems;
    private final int maxNumberOfItems;

    public GroceryList() {

    public GroceryList(int maxNumberOfItems) {

    public boolean addItem(GroceryItem item) {

    public boolean removeItem(String name) {

    public double getTotalCost() {

    public String toString() {

}
```

JOptionPane

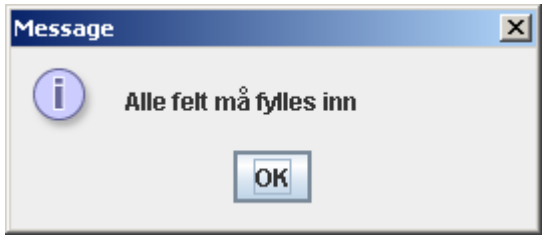
En option pane er en enkel dialogboks for grafisk input/output.



- fordeler:
  - enkle
  - fleksible (på mange måter)
  - ser ok ut
- ulemper:
  - lages med static metoder – ikke veldig objektorientert
  - ikke veldig 'kraftige' (bare enkle dialogbokser)

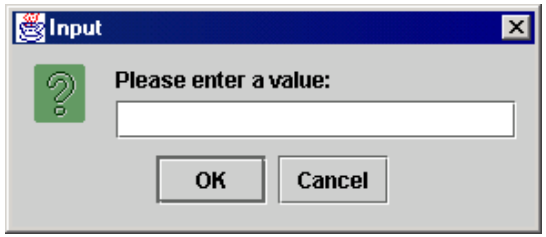
Typer

- `showMessageDialog(`***<parent>, <message>***`)`
  Viser en melding med en OK-knapp.



- `showConfirmDialog(`***<parent>, <message>***`)`
  Viser en melding og en liste med valg:
  Yes, No, Cancel;
  Returnerer brukerens valg som en `int` med en av følgende verdier:
  - `JOptionPane.YES_OPTION`
  - `JOptionPane.NO_OPTION`
  - `JOptionPane.CANCEL_OPTION`

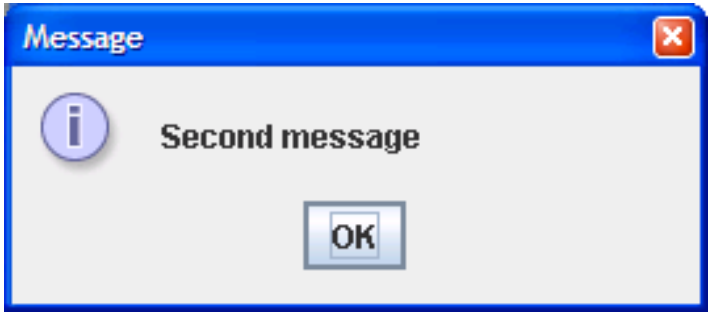
- `showInputDialog(`***<parent>, <message>***`)`
  Viser en melding og et tekstfelt for input.
  Returnerer brukers verdi som en `String`.
  - `null` kan brukes som "parent" for alle metodene



Eksempel 1

- `showMessageDialog` tilsvarer `System.out.println` for å vises en melding:

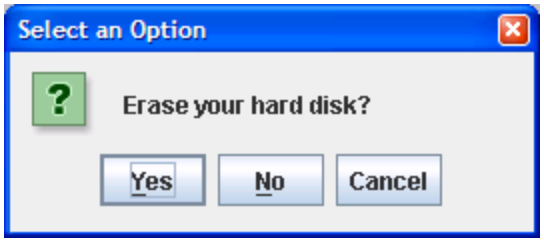
```
import javax.swing.*;

public class MessageDialogExample {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(
                        null,"How's the weather?");
        JOptionPane.showMessageDialog(
                        null, "Second message");
    }
}
```

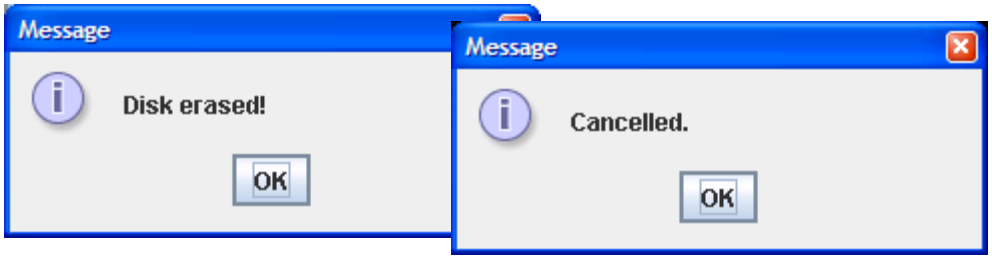


Eksempel 2

- `showConfirmDialog` tilsvarer `System.out.print` som skriver et spørsmål og som leser input fra bruker (ett av de mulige valgene)

```
import javax.swing.*;

public class ConfirmDialogExample {
    public static void main(String[] args) {
        int choice = JOptionPane.showConfirmDialog(
                                        null, "Erase your hard disk?");
        if (choice == JOptionPane.YES_OPTION) {
            JOptionPane.showMessageDialog(null, "Disk erased!");
        } else {
            JOptionPane.showMessageDialog(null, "Cancelled.");
        }
    }
}
```

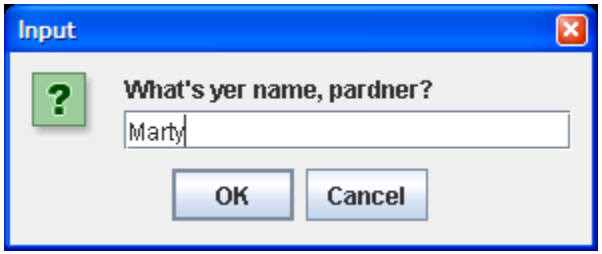




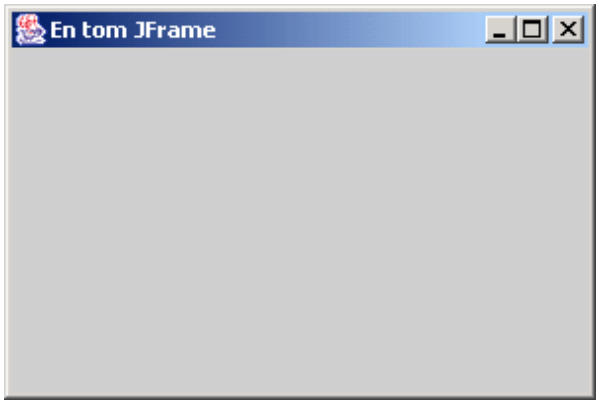
Eksempel 3

- `showInputDialog` tilsvarer `System.out.print` som skriver et spørsmål og som leser input fra bruker (hva som helst)

```
import javax.swing.*;

public class InputDialogExample {
    public static void main(String[] args) {
        String name = JOptionPane.showInputDialog(null,
                                    "What's yer name, pardner?");
        JOptionPane.showMessageDialog(null, "Yeehaw, " + name);
    }
}
```



javax.swing.JFrame - Vinduet



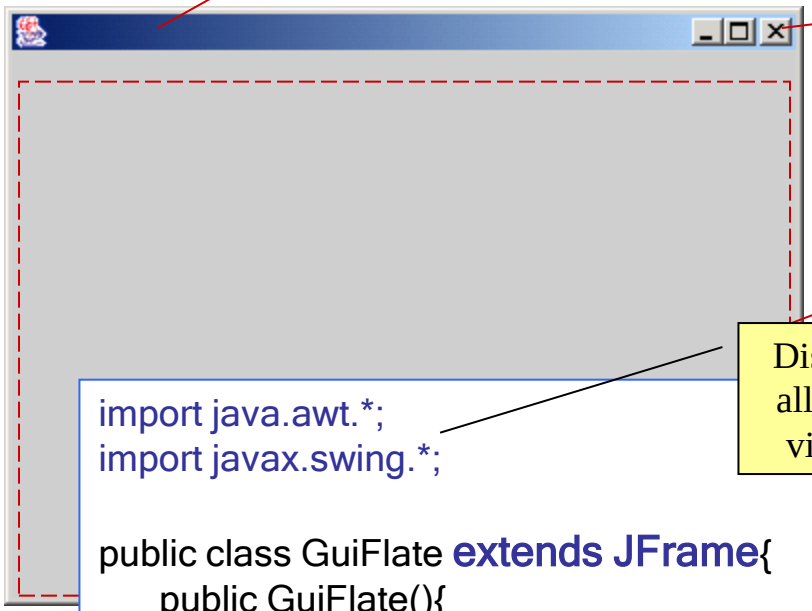
- Vinduet er en **beholder** (Container) der man kan legge inn ulike gui-komponenter.
- En klasse kan definere et vindu ved å arve fra klassen **JFrame** (som er et "tomt" vindu)
- Konstruktøren i denne sub-klassen brukes til å *bygge* GUI'en
  - instansierer ulike gui-objekter og legger dem inn i vinduet, eller rettere: på vinduets arbeidsflate (”contentpane”)

Et vindu uten innlagte gui-komponenter

tittelfelt (her uten tittel)

minimering, maksimering

Vindu (JFrame)



guiFlaten: det er her vi ønsker å legge til gui-komponenter

Disse to pakkene må alltid importeres når vi jobber med GUI

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

public class GuiFlate extends JFrame{
    public GuiFlate(){

        //legge til gui-komponenter (knapper, tekst...)
        //...
        setSize(300, 200);  //standard: ingen utstrekning
        setVisible(true);//standard: usynlig
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    ...
}
```

Heltalls-konstant

GUI – skritt for skritt

- Når man ”bygger gui” vil man merke et gjentakende mønster av steg som må utføres

Som attributter (datafelt/objektvariabler) i klassen:
1) Deklarere (gui-)komponenter

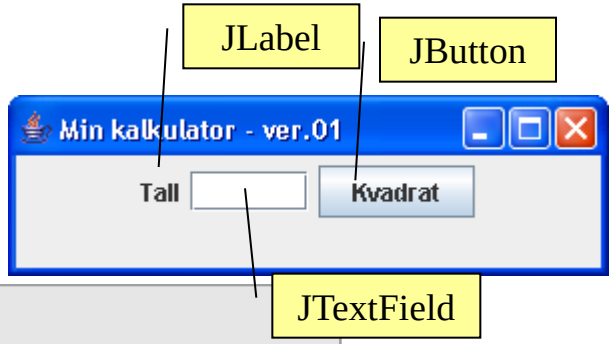
i kontruktøren:
2) **organisere** arbeidsflaten (setLayout(..))
3) **opprette** komponentene (new...)
4) **legge til** komponentene til arbeidsflaten (add(..))
5) Sette størrelse, synliggjøre, og avslutningsprosess

i klient (main-metode)
6) Instansiere GUI-vindu fra main-metoden

Med hendelseshåndtering (dvs. funksjonalitet):
3b) lage **lytterklasser** (implements xxxListener)
3c) **koble** gui-komponenter til lyttere (addxxxListener)

API for noen GUI-komponenter

- Konstruktører



Definert i klasse	Konstruktør/metode	Beskrivelse
Jlabel	**JLabel**(String text) **JLabel**(String text, int horizontalAlignment)	Oppretter en etikett med angitt tekst Som over, samt horisontal-justering (venstre, høyre, center).

Definert i klasse	Konstruktør/metode	Beskrivelse
JButton	**JButton**() **JButton**(String text) **JButton**(String text, Icon icon)	Oppretter knapp med henholdsvis ingen tekst, spesifisert tekst og spesifisert tekst med et ikon.

Definert i klasse	Konstruktør/metode	Beskrivelse
JTextField	**JTextField**() **JTextField**(int columns) **JTextField**(String text) **JTextField**(String text, int columns)	Oppretter et tomt tekstfelt. Oppretter et tomt tekstfelt med angitt bredde. Oppretter tekstfelt med angitt tekst. Oppretter tekstfelt med angitt tekst og bredde.

Eksempel1.java

```
import java.awt.*;  //Container, FlowLayout
import javax.swing.*; //JFrame, JLabel, JTextArea, JButton

public class Eksempel1 extends JFrame {
  private JButton btnOk;
  private JTextField txtNavn;

  public Eksempel1() {
    setTitle("Første GUI");
    setLayout(new FlowLayout());
    txtNavn = new JTextField(15);
    btnOk = new JButton("OK");
    add(new JLabel("Skriv navnet ditt"));
    add(txtNavn);
    add(btnOk);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setSize(400, 200);
    setVisible(true);
  }
  public static void main(String[] args) {
    new Eksempel1();
  }
}
```

Prefiks på referanser til gui-objekter

- Det er alltid hensiktsmessig å gjøre kildekode mest mulig leselig.
- Foruten formatering av tekst (innrykk, blokker m.m) øker bruk av **prefiks på variabelnavn** til gui-komponenter "*lesbarheten*" av kildekode betraktelig (vet hvilken type gui-komponent)

GUI-komponent	Prefiks
JLabel	lbl
JButton	btn
JTextField	txt
JPasswordField	psw
JTextArea	txa
JScrollPane	jsp
JCheckBox	chk
JRadioButton	rbt
JPanel	pnl

Hvordan få *funksjonalitet* i programmet?

- Programmet er dødt slik det er nå.
- Hvordan få programmet til å ”gjøre noe” dersom f.eks. en knapp trykkes?

1. Det må defineres en klasse som implementerer et ”lytter-interface”, som gjør objekter av klassen i stand til å "**lytte**" (eng: listen) til en viss type hendelser (eng: event).
   – hvilket lytter-interface klassen skal implementere avhenger av hvilken *type hendelse* objekter av klassen skal kunne lytte etter.
2. Instansierte lytter-objekter må **registreres** med den gui-komponent som vi ønsker å fange opp hendelser for.

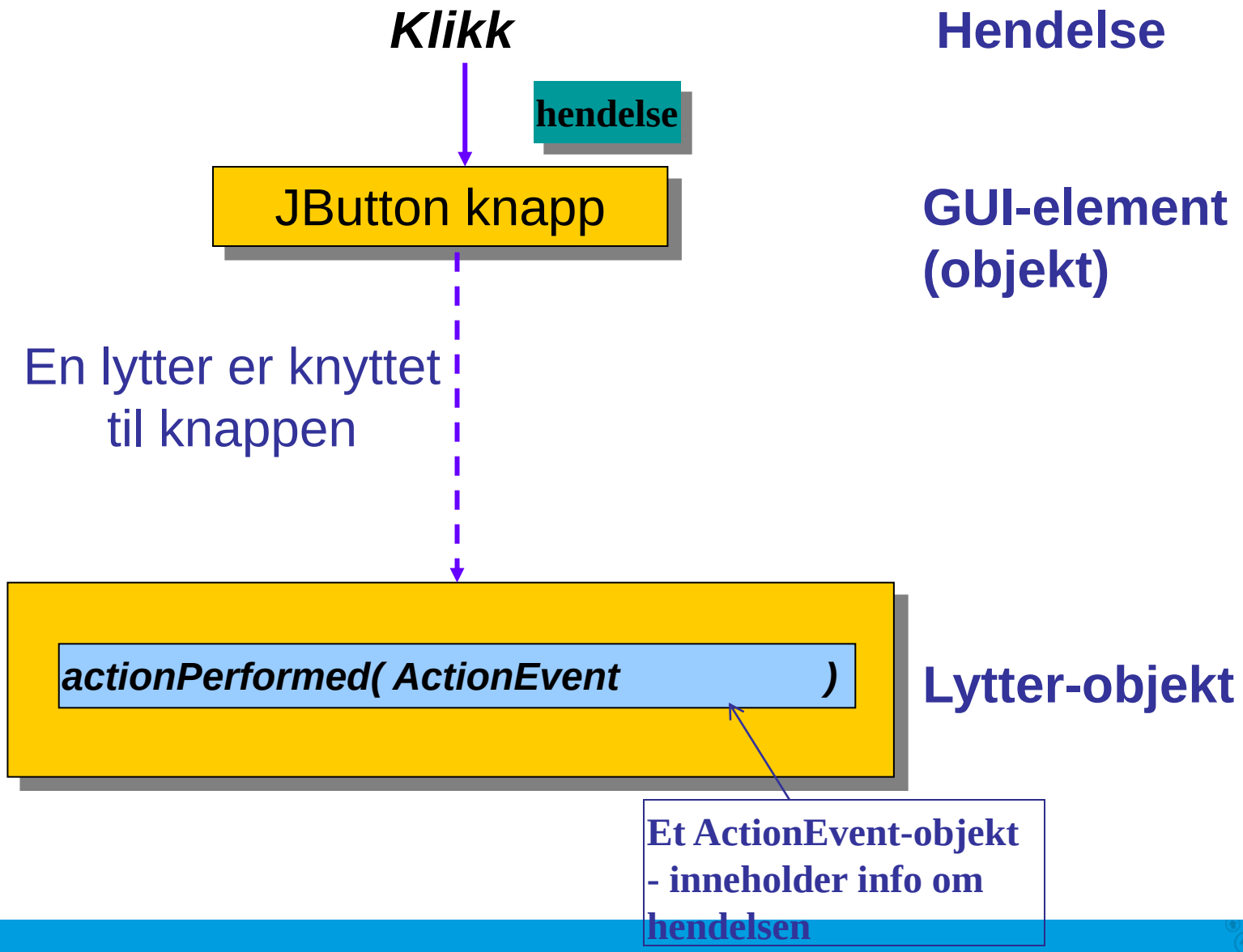
Tilsvarer 3b) og 3c) i ”GUI - skritt for skritt”

Hva er “Hendelser”

- Programflyten i et GUI-program er styrt av *hendelser*.
  - En knapp som *trykkes*
  - Det slås <enter> i et tekstfelt.

  → **ActionEvent**
  - Et tekstfelt *forlates* (med <tab>) eller *ankommes* → **FocusEvent**
  - Musepeker som flyttes → **MouseEvent**
  - Vindu som endrer størrelse, lukkes, minimeres… → **WindowEvent**
  - Mange flere…

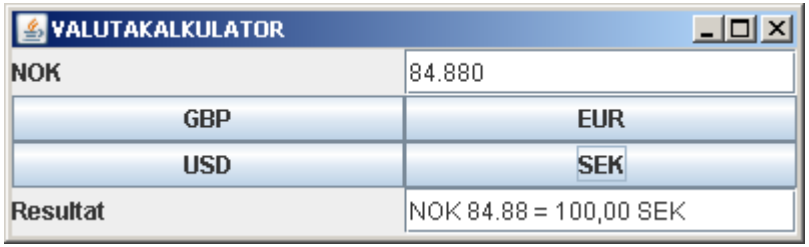
Hendelseskontroll



Eksempel 1 (revidert)

```
import java.awt.*;
import javax.swing.*;
public class Eksempel1 extends Jframe implements ActionListener {
   private JButton btnOk;
   private JTextField txtNavn;
   public Eksempel1() {
      setTitle("Første GUI");
      txtNavn = new JTextField(15);
      txtNavn.addActionListener(this);
      btnOk = new JButton("OK");
      btnOk.addActionListener(this);
      add(new JLabel("Skriv navnet ditt"));
      add(txtNavn);
      add(btnOk);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setSize(400, 200);
      setVisible(true);
   }
   public static void main(Sring[] args) {
      new Eksempel1();
   }
   public void actionPerformed(ActionEvent event) {
      String navn = txtNavn.getText();
      JOptionPane.showMessageDialog(this, navn + " er registrert!");
   }
}
```

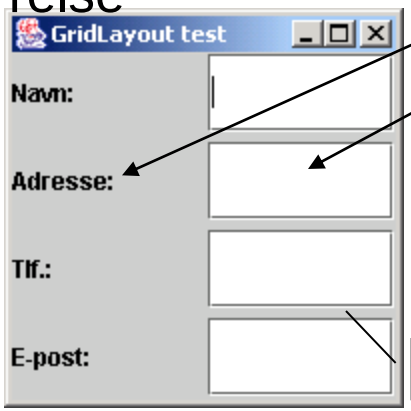
Eksempel 2 - Valutakalkulator.java




```
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Valutakalkulator extends JFrame implements ActionListener {
    //attributter
    private JButton btnGbp, btnEur, btnUsd, btnSek;
    private JTextField txtNok, txtResultat;
    private JLabel lblNok;
    private final double GBP = 8.873;
    private final double USD = 5.6;
    private final double EUR = 7.498;
    private final double SEK = 84.880;

    public Valutakalkulator() {

    public static void main(String[] args) {

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

GridLayout

- **Rekkefølgen** for innlegging har betydning for plassering av komponenter.
- Ingen ”celler” kan hoppes over.
- Like store komponenter
- Beholder innbyrdes plassering dersom vinduet endrer størrelse



"Luft": 5 pixler

```
//import...
public class TestGridLayout extends JFrame{
  private JTextField txtNavn = new JTextField(15);
  private JTextField txtAdresse = new JTextField(15);
  private JTextField txtTlf = new JTextField(15);
  private JTextField txtEPost = new JTextField(15);

  public TestGridLayout() {
    super("GridLayout test");
    setLayout( new GridLayout(4, 2, 5, 5) );
    add(new JLabel("Navn:"));
    add( txtNavn );
    add(new JLabel("Adresse:"));
    add( txtAdresse );

    add(new JLabel("Tlf.:"));          add( txtTlf );
    add(new JLabel("E-post:"));    add( txtEPost );
    setSize(200, 200);
    setVisible(true);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public static void main(String[] args){
              new TestGridLayout();
  }
}
```

OK å instansiere GUI-komp. før konstruktør.

luft

GridLayout()
Creates a grid layout with a default of one column per component, in a single row.
GridLayout(int rows, int cols)
Creates a grid layout with the specified number of rows and columns.
GridLayout(int rows, int cols, int hgap, int vgap)
Creates a grid layout with the specified number of rows and columns.