agneshb@uio.no
IN1010 gruppe 3

GUI 2

I DAG

Ulike typer layout

Designegenskaper ved GUI

MVC

Oppgaver

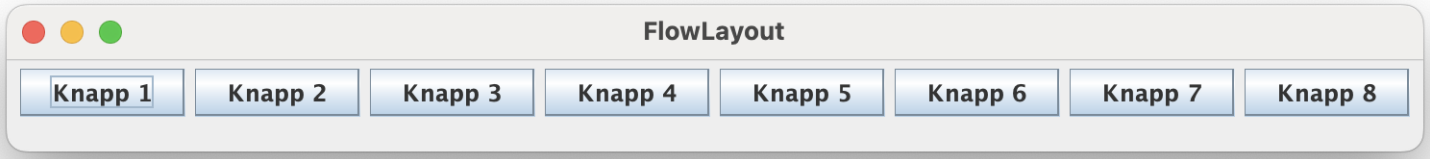
LAYOUT

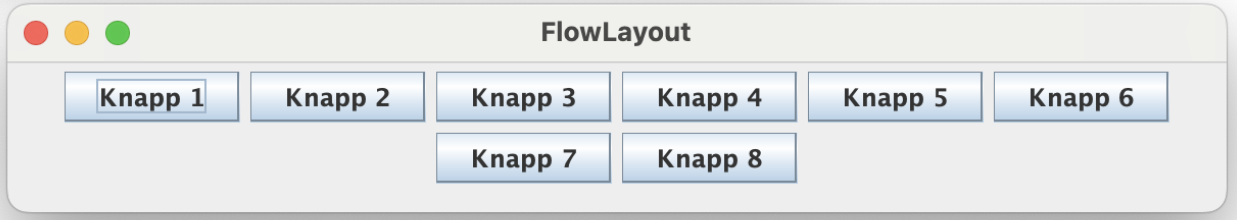
- Avgjør hvordan elementer plasseres JPanel-et.
- FlowLayout
- GridLayout
- BorderLayout

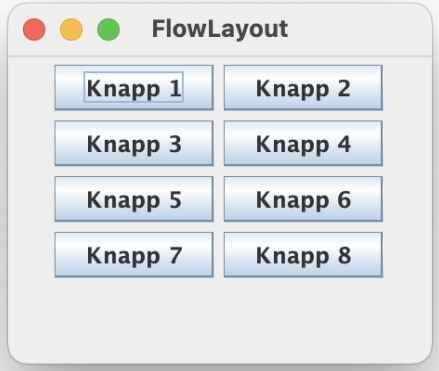
FLOW LAYOUT

- Standard
- Elementene legges etter hverandre i rekkefølgen de blir lagt til
- «Wrapper» for å få plass til alt i vinduet
- Som en teksboks PowerPoint el.l. som du kan endre form på → ordene flytter seg etter plassen i boksen.



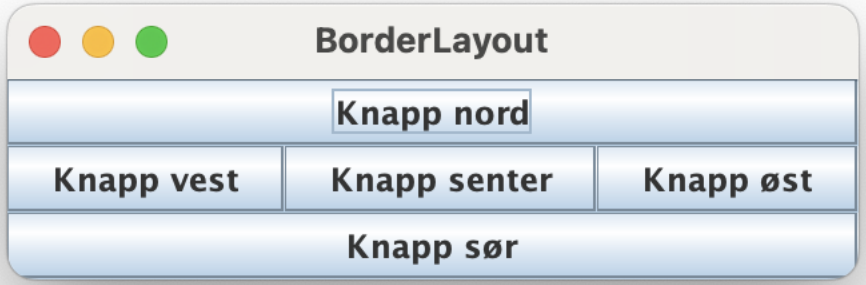


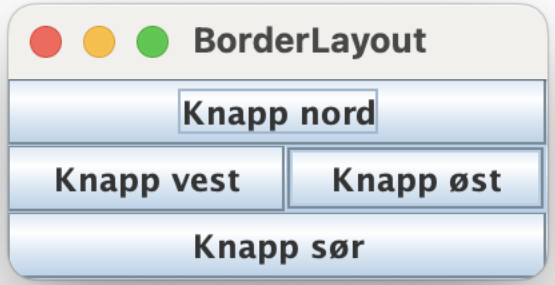




BORDER LAYOUT

- Velger en posisjon for elementet: NORTH, SOUTH, EAST, WEST, CENTER

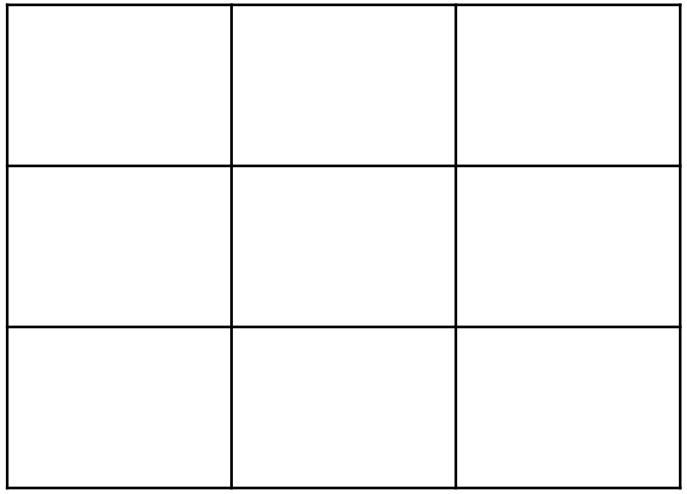


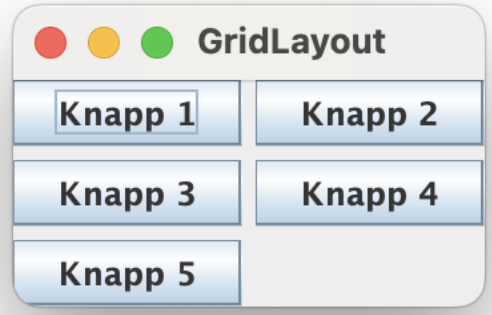


Fyller opp plassen hvis ikke alle posisjonene er tatt

GRID LAYOUT

- Lager et rutenett med antall rader og kolonner
- Elementene legges til i neste ledige celle





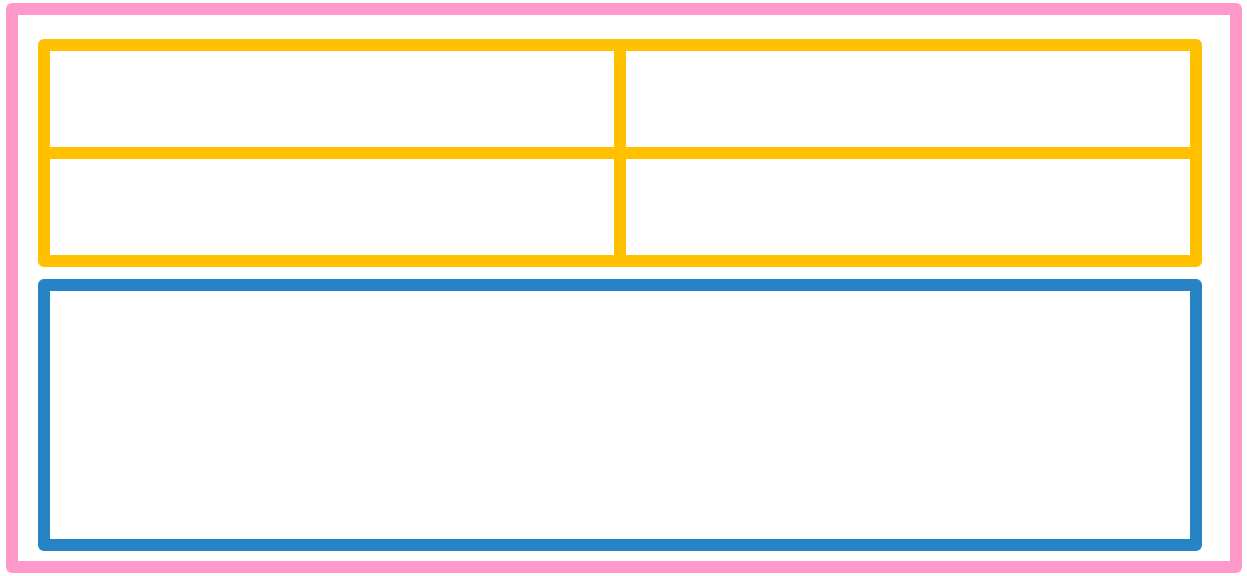
GridLayout med 3 rader og 2 kolonner

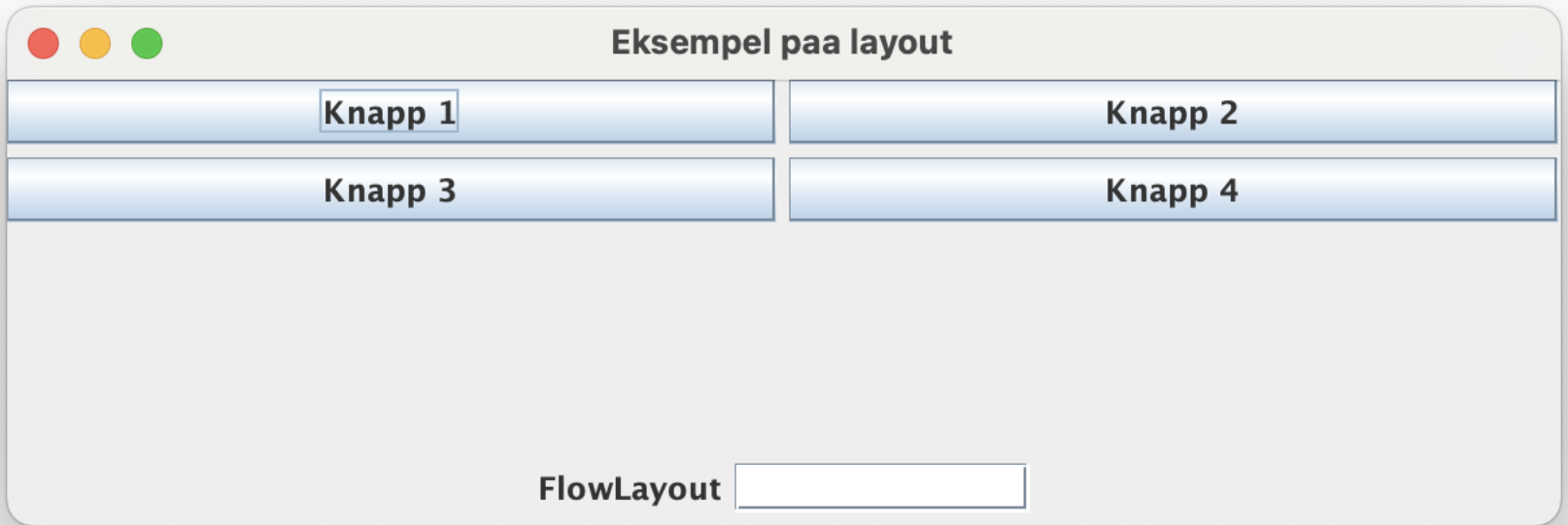
FLERE LAYOUTS SAMMEN

Ofte kombinerer man flere type layouts for å få ønsket resultat

Se kode Layouteksempel.java

BorderLayout med en
GridLayout i nord og en
FlowLayout i sør.




Eksempel paa layout
Knapp 1
Knapp 2
Knapp 3
Knapp 4
FlowLayout

EGENSKAPER VED ULIKE GUI-OBJEKTER

ActionEvent
interface ActionListener
- actionPerformed()

BorderFactory
- createLineBorder(Color c)

BorderLayout ext LayoutManager
- String CENTER, EAST, NORTH, SOUTH, WEST

Color
- Color BLACK, BLUE, ...

Dimension
FlowLayout ext LayoutManager
Font
- int BOLD, PLAIN
- String MONOSPACED, SANS_SERIF

GridLayout ext LayoutManager
JButton ext JComponent
- addActionListener (ActionListener a)

JComponent
- setBackground(Color c)
- setFont (Font f)
- setForeground(Color c)
- setOpaque(boolean b)
- setText (String t)

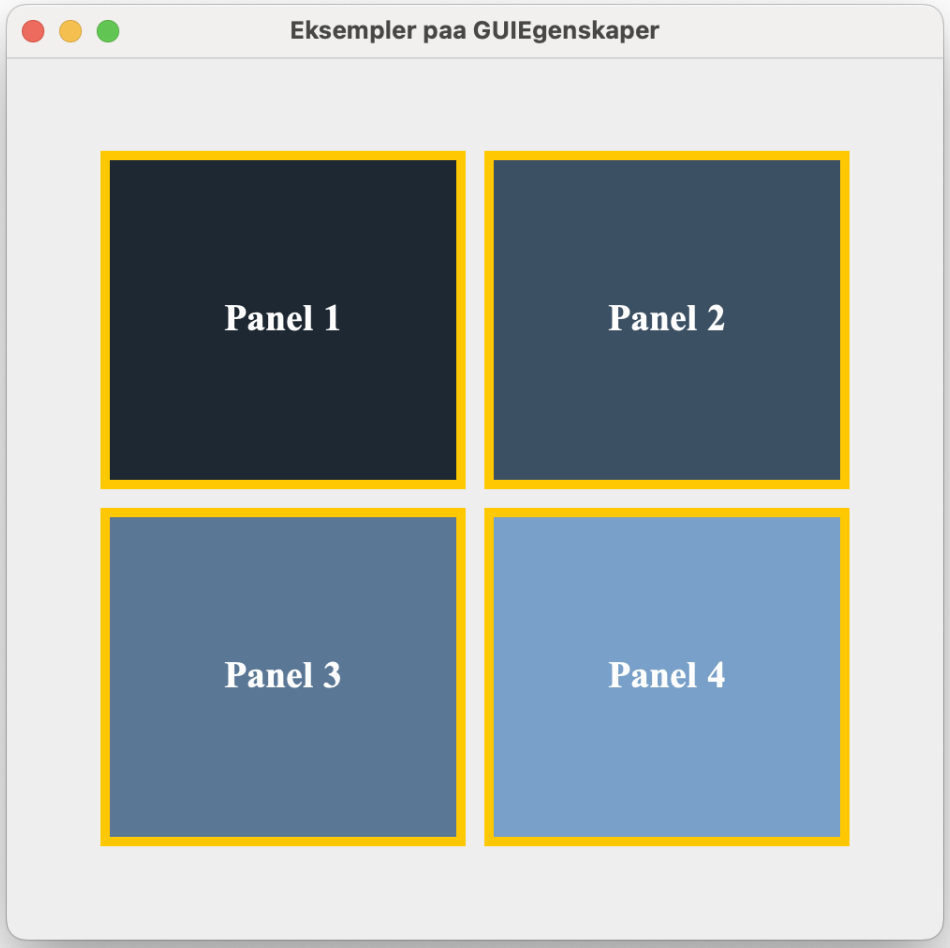
JFrame
- add (JComponent c)
- int EXIT_ON_CLOSE
- pack()
- setDefaultCloseOperation (int i)
- setVisible (boolean b)

JLabel ext JComponent
JPanel ext JComponent
- add (JComponent c)
- setLayout(LayoutManager m)

interface LayoutManager

EKSEMPEL

- Sentrere tekst
- Angi tekstfarge, teksttype og bakgrunnsfarger
- Legge til kant/border rundt rutene
- Legge til marger mellom rutene
- Legge til marg rundt hele griden



MODEL-VIEW-CONTROLLER (MVC)

- Designmønster som brukes til utvikling av brukergrensesnitt
- Deler opp programmet vårt i ansvarsområder:
  - **Model**: håndterer dataen til applikasjonen
  - **View**: brukergrensesnittet
  - **Controller**: håndterer brukerinteraksjon fra viewet og endrer/oppdaterer data i modellen

HVORFOR BRUKE MVC?

- Høy kohesjon
  - Samle objektegenskaper som hører sammen
  - Grafiske deler og datahåndtering er i separate klasser
- Lav kobling
  - objektene avhenger av og har referanser til så få andre objekter som mulig
  - view ikke har referanser til både controller og model
- Enklere å vedlikeholder og utvide

MVC REFERANSER

Model: ikke referanse til view eller controller

View: referanse til controller, ikke til model

Controller: referanse til view og model

