GUI 2

IN1010 -GRUPPE 2

13.mai 2025

this week>

- Layout eksempler
  - Flowlayout
  - GridLayout
  - BorderLayout
- Designegenskaper til GUI
- MVC
- seminaroppgaver

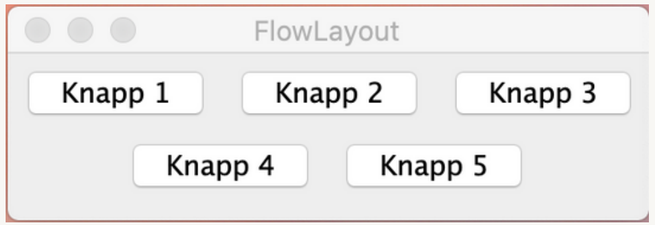
Layout Manager

- **En layout Manage**r organiserer/strukturer det visuelle innholdet på en gitt måte
  - f.eks hvordan knapper og tekst skal plasseres i en JPanel
  - FlowLayout
  - GridLayout
  - BorderLayout

Layout Manager

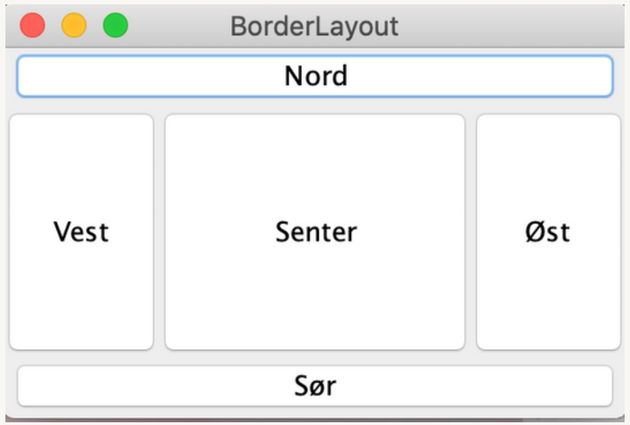
FlowLayout:

- knappene ligger etter hverandre horisontal



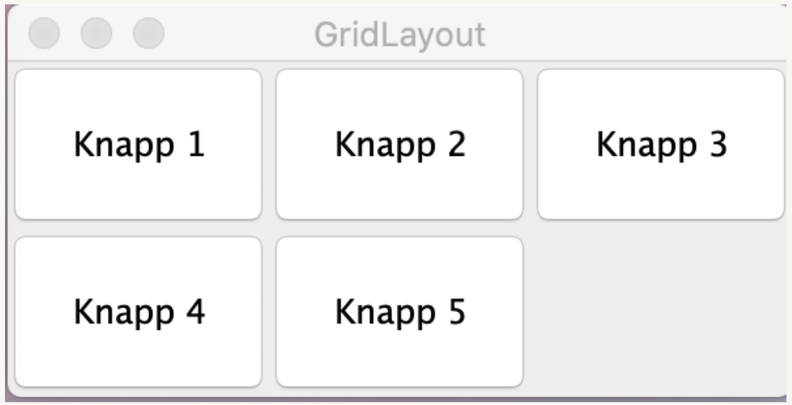
BorderLayout:

- Knappene blir delt inn i forskjellige områder, sør, øst,vest, nord og senter



GridLayout:

- knappene ligger i 2x3 rutenett



Layout Manager

- Bruker metoden .setLayout() på en JFrame-objekt
- sender inn instans av FlowLayout(), BorderLayout(), GridLayout() etc

```
flowFrame.setLayout(new FlowLayout());

for (int i = 1; i <= 5; i++) {
    flowFrame.add(new JButton("Knapp " + i));
}
```

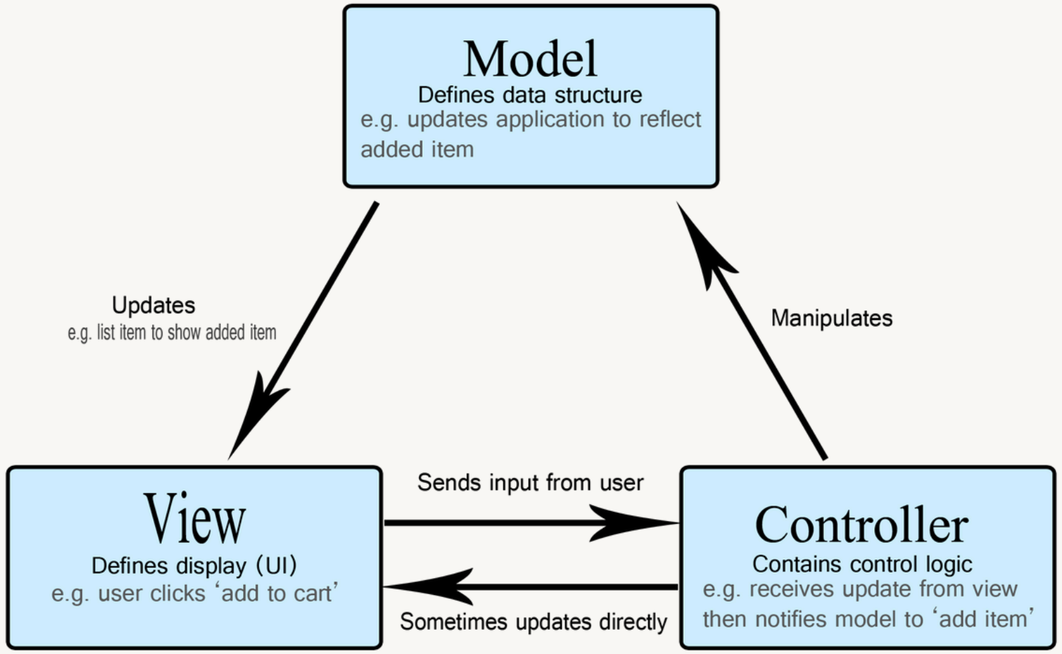
Designegenskapene til GUI

- Vi kan legge til bakgrunnsfarge, teksttyper, tekststørrelse, tekstfarge og mye mer
- Klasser og metoder som har blitt nevnt i forelesning:

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

Model view control (MVC)

- Et designmønster som deler opp programmet i ansvarsområder
  - Datahåndtering → **model**
  - Brukergrensesnitt → **view**
  - Fordeling av ansvar mellom model og view → **controller**



Hvorfor MVC?

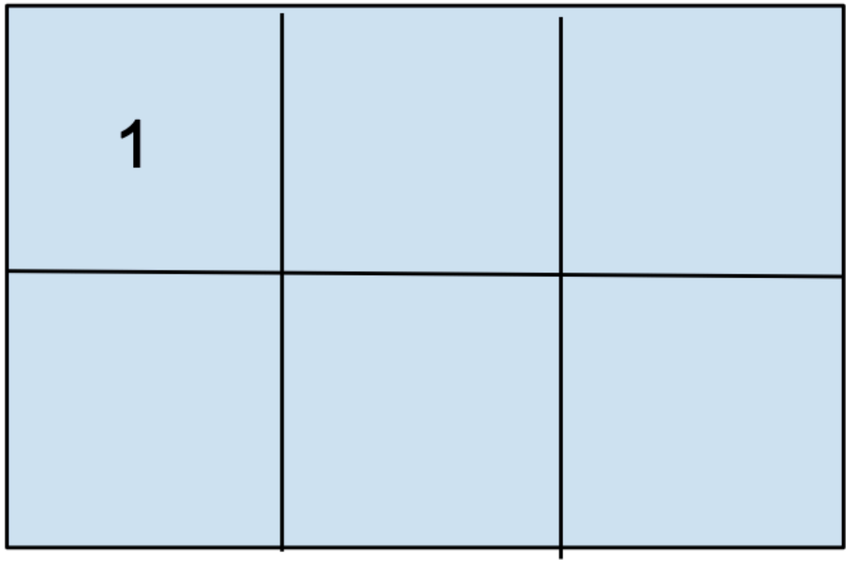
- I objektorientert programmering ønsker vi en høy grad av sammenheng (**cohesion**) med færrest mulig objekter(**loose coupling**)
- Blir enklere å vedlikeholde og utvidet over tid
- **Høy cohesion**
  - Samle objektegenskapene hver for seg
  - f.eks databehandling og brukergrensesnitt
- **Loose coupling**
  - La objekter ha færrest mulig referanser til andre objekter
  - f.eks ikke la view ha referanse til både model og view

MVC eksempel: handleliste

```
class ShoppingListApp {
    Run | Debug
    public static void main(String[] args) {
            ShoppingListModel model = new ShoppingListModel();
            ShoppingListView view = new ShoppingListView();
            ShoppingListController controller = new ShoppingListController(model, view);
            view.setController(controller);
    }
}
```

Oppgave 1: Grid med tall- og smileknapper

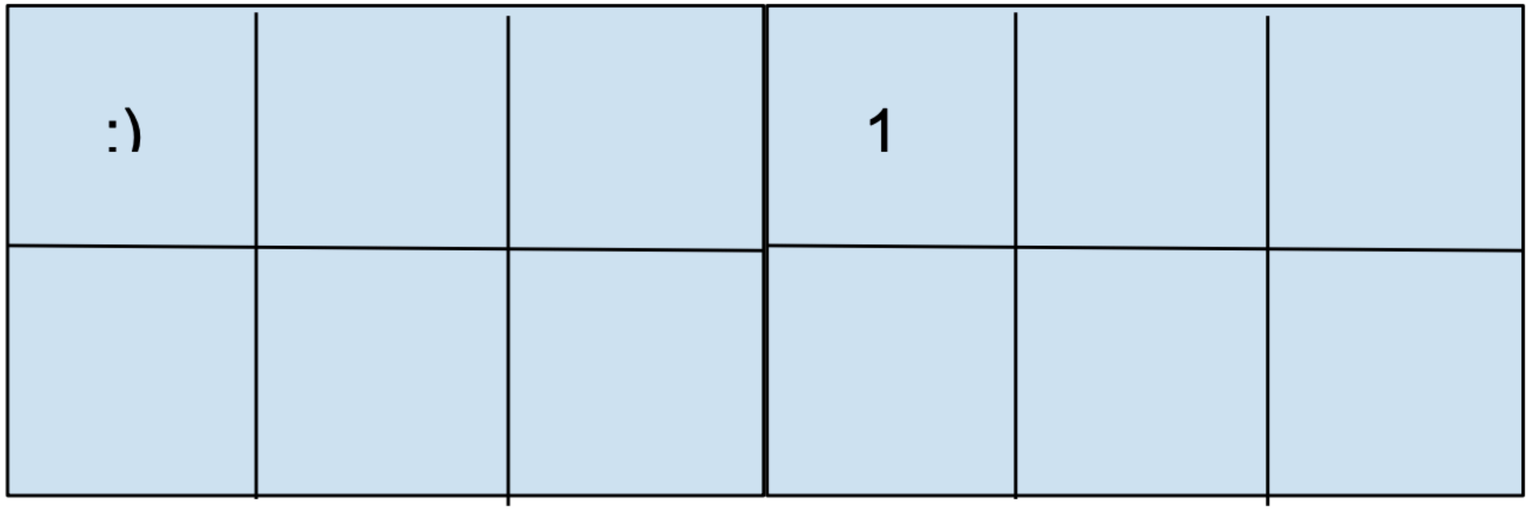
a) Lag et 2x3 grid med seks ruter. Hver rute skal ha en knapp, og det ser da slik ut:



b) Vi ønsker at knappene skal være mulige å trykke på. Når en knapp blir trykket på skal tallet på knappen skiftes ut med et :). Hint: Her kan det være lurt å lage en subklasse til JButton. Hvis jeg nå trykker på knapp 1 og 5 vil griden nå se slik ut:



c) Endre til at det veksles mellom et smilefjes og tallet for hver gang knappen trykkes. Eksempel: trykker på knapp 1 og 5, som får smilefjes (venstre). Trykker deretter på knapp 1 og 5 igjen, som får tilbake sine originale tall (høyre).



seminaroppgaver