IN1010 Gruppetime – Uke 13

Repetisjon

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

class GUIEksempel {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        JFrame vindu = new JFrame("Vindu-tekst");
        vindu.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        vindu.setPreferredSize(new Dimension(500, 300));

        try {
            UIManager.setLookAndFeel(
                UIManager.getCrossPlatformLookAndFeelClassName()
            );
        } catch (Exception e) {
            System.exit(1);
        }

        vindu.pack();
        vindu.setLocationRelativeTo(null);
        vindu.setVisible(true);
    }
}
```



Noen objekter

- JFrame: Vinduet som opprettes, og som brukeren ser.
- JPanel: En 'tegneflate' som andre objekter kan legges inn i. Man kan bestemme hvordan objektene skal legges inn (layouts).
- JLabel: Et objekt med tekst, som vi kan legge inn i et panel.
- JButton: En trykkbar knapp. Vi kan spesifisere handling som skjer når knappen trykkes på.
- JTextField: Et tekstfelt. Bruker kan skrive hva som helst i feltet.
- Etc etc…

```java
JFrame vindu = new JFrame("Vindu-tekst");
vindu.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
vindu.setPreferredSize(new Dimension(500, 300));

try {
    UIManager.setLookAndFeel(
        UIManager.getCrossPlatformLookAndFeelClassName()
    );
} catch (Exception e) {
    System.exit(1);
}

JPanel panel = new JPanel();
JLabel tekst = new JLabel("Dette er tekst");
JButton button = new JButton("Dette er en knapp");

panel.add(tekst);
panel.add(button);
vindu.add(panel);

vindu.pack();
vindu.setLocationRelativeTo(null);
vindu.setVisible(true);
```



Layout for JPanel

- Hvis ingenting annet er spesifisert, så vil elementer i JPanel legges til etter hverandre.
- Man kan spesifisere layout til JPanel, til hva man vil ha.
  - BorderLayout: Man kan plassere elementer nord, sør, øst, vest eller i midten
  - GridLayout: Spesifiserer hvor mange rader og kolonner man vil ha, elementer vil legges til i et rutenett
  - FlowLayout: Elementer legges til i rad, så lenge det er plass

```java
JPanel panel = new JPanel(new BorderLayout(10, 10));
JButton nknapp = new JButton("Nord");
JButton sknapp = new JButton("Sør");
JButton vknapp = new JButton("Vest");
JButton øknapp = new JButton("Øst");
JButton mknapp = new JButton("Midt på");

panel.add(nknapp, BorderLayout.NORTH);
panel.add(sknapp, BorderLayout.SOUTH);
panel.add(vknapp, BorderLayout.WEST);
panel.add(øknapp, BorderLayout.EAST);
panel.add(mknapp, BorderLayout.CENTER);

vindu.add(panel);
```



Hendelser

- Akkurat nå gjør knappene ingenting når de blir trykket på
- For å bestemme handlinger, kan vi definere en klasse som implementerer grensesnittet *ActionListener*
- Denne klassen venter på en hendelse (f.eks trykk av en knapp), og kjører metoden *actionPerformed* når hendelsen skjer.

```java
static int count = 0;
```

```java
JPanel panel = new JPanel();
JLabel tekst = new JLabel("Telling: " + count);
JButton knapp = new JButton("Øk teller");
knapp.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        count++;
        tekst.setText("Telling: " + count);
    }
});

panel.add(tekst);
panel.add(knapp);

vindu.add(panel);
```





MVC-Mønsteret

- Tilbakeblikk til designmønstre
- MVC er et mønster som kan brukes for å gjøre GUI-kode mer oversiktlig og lettere å implementere
- Består av 3 deler
  - Modell: Inneholder backend-logikk og data vi er interessert i
  - View: Viser brukergrensesnittet og håndterer brukerinteraksjon. Dette er det bruker ser.
  - Controller: Karen i midten, kobler view og model. Sørger for at ting i begge klassene oppdateres samtidig.