



PGR 101 Objektorientert Programmering 2  
Vår 2017

# Forelesning 18.4.17

(Stein Marthinsen – [marste@westerdals.no](mailto:marste@westerdals.no))

# Dagens tema

Mer Grafisk Grensesnitt (GUI)

Forrige øvingsoppgave

`JOptionPane`

`MessageDialog`

`InputDialog`

`ConfirmDialog`

`JLabel`

`JTextField`

Konvertere tekst til tall

Hente fra tekstfelt/legge ut i tekstfelt/label

# Kan du dette?

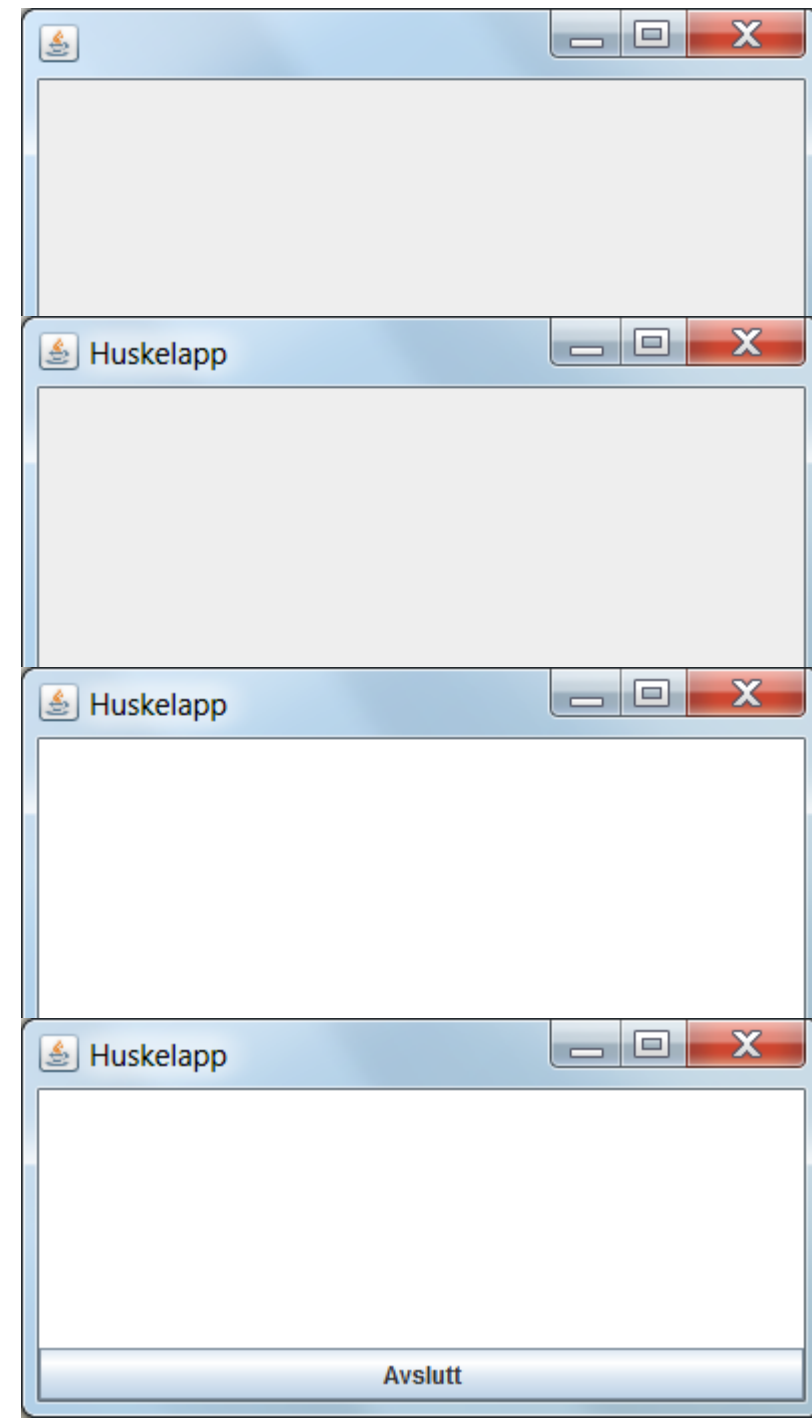
Sett opp et vindu (JFrame)

Sette tittel på vinduet

Legge et tekstområde i vinduet

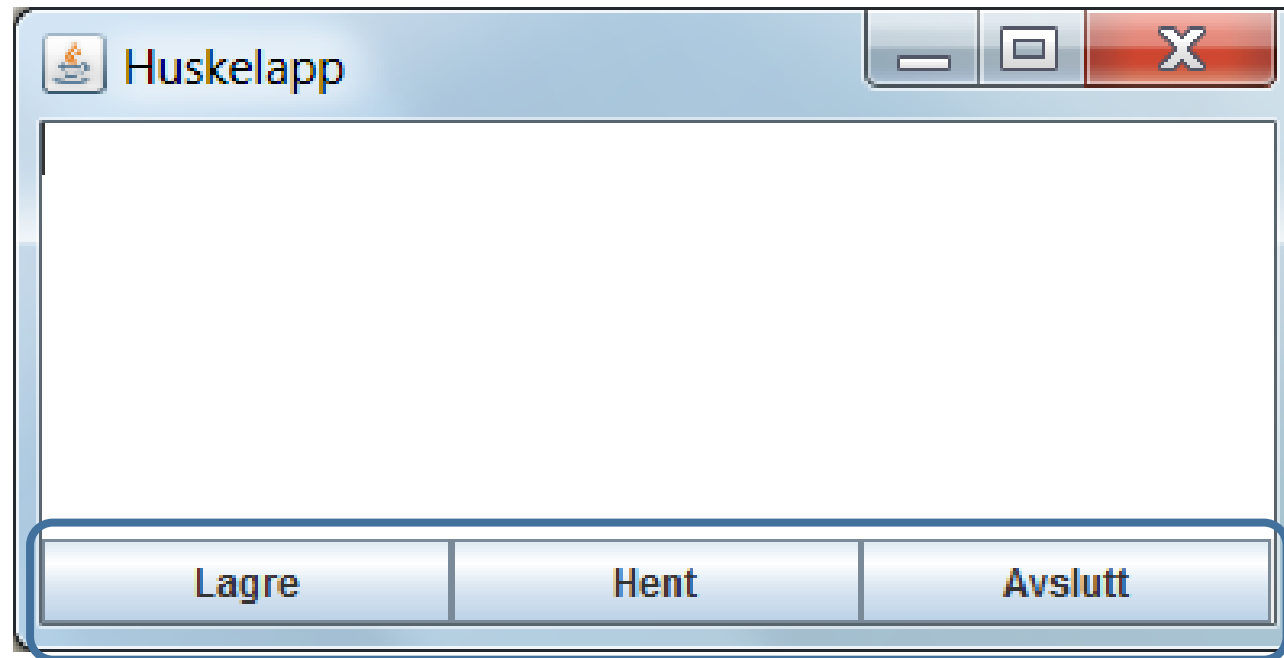
Bruke BorderLayout

Legge en knapp i vinduet



# Bruk av paneler

Med et panel, kan du legge inn komponenter med en valgfri layout. Og så plassere panelet i vinduet.



Dette er nå  
et panel

```
public Note() {  
    setTitle("Huskelapp");  
    add(new JTextArea(), BorderLayout.CENTER);  
    JPanel pnlSouth = new JPanel(new GridLayout(1, 3));
```

Et panel med 1 rad og 3 kolonner

--	--	--

```
add(pnlSouth, BorderLayout.SOUTH);
```



# Forrige øving

Lagre

Hent

Avslutt

```
public void actionPerformed(ActionEvent e) {  
    String clicked = e.getActionCommand();  
    if ("Hent".equals(clicked)) {  
        System.out.println("Du valgte Hent");  
    } else  
    if ("Lagre".equals(clicked)) {  
        System.out.println("Du valgte Lagre");  
    } else {  
        System.exit(0);  
    }  
}
```

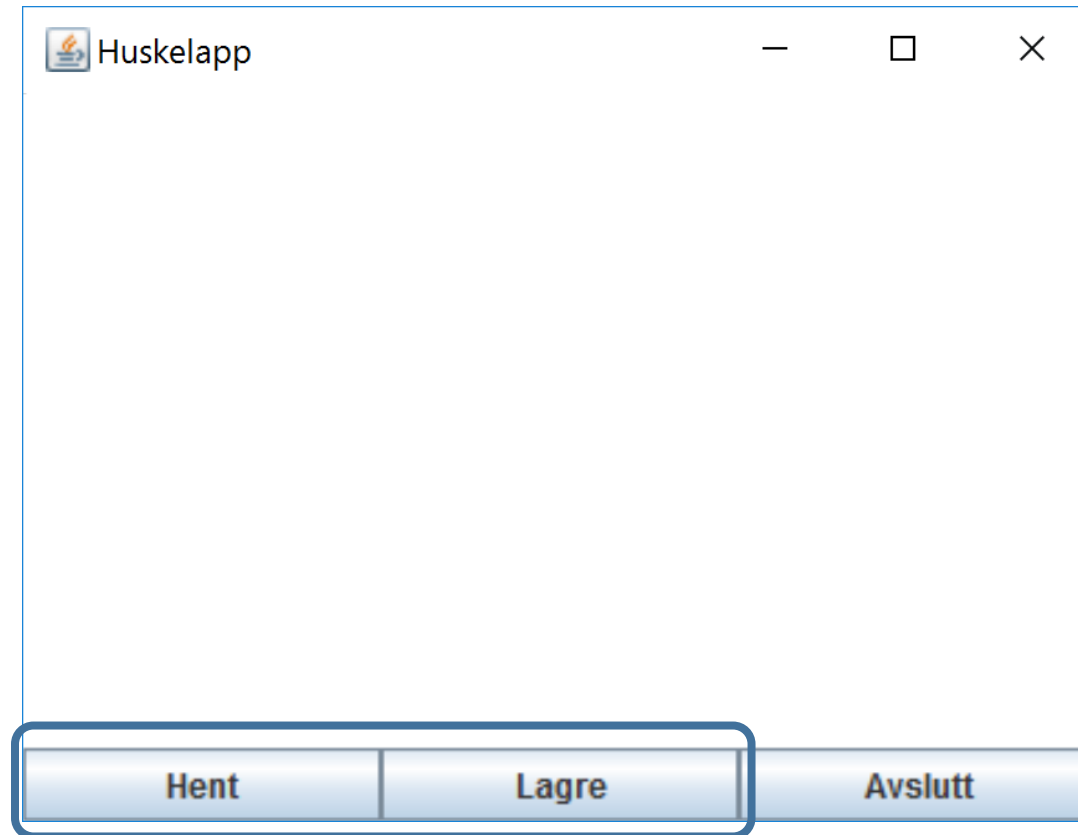
Henter teksten  
på knappen

# Forrige øving - alternativ

```
public void actionPerformed(ActionEvent e) {  
    Object clicked = e.getSource(); Henter kilden til hendelsen  
    if (clicked == btnOpen) {  
        System.out.println("Du valgte Hent");  
    } else  
    if (clicked == btnSave) {  
        System.out.println("Du valgte Lagre");  
    } else {  
        System.exit(0);  
    }  
}
```

# Huskelapp

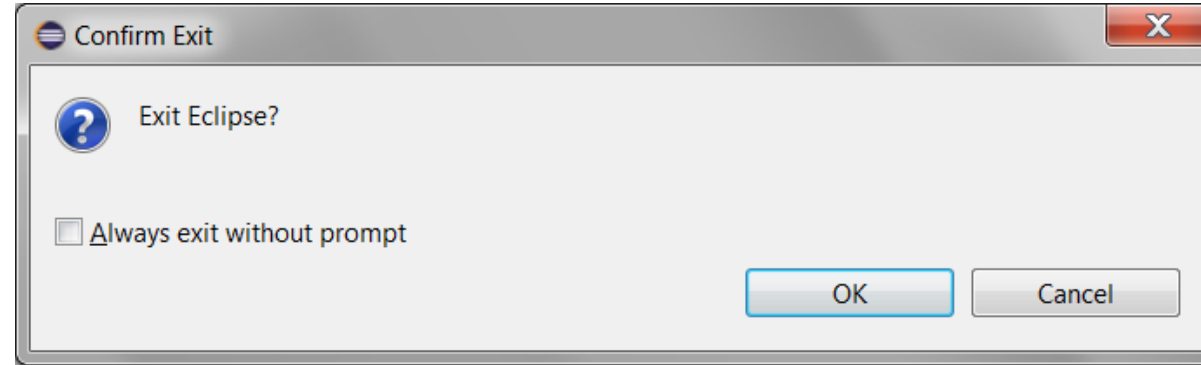
Vi skal snart gjøre denne applikasjonen ferdig!



Dette krever filbehandling!



# Dialogvinduer



## BankID på mobil

### Logg inn

Mobilnummer (8 siffer)

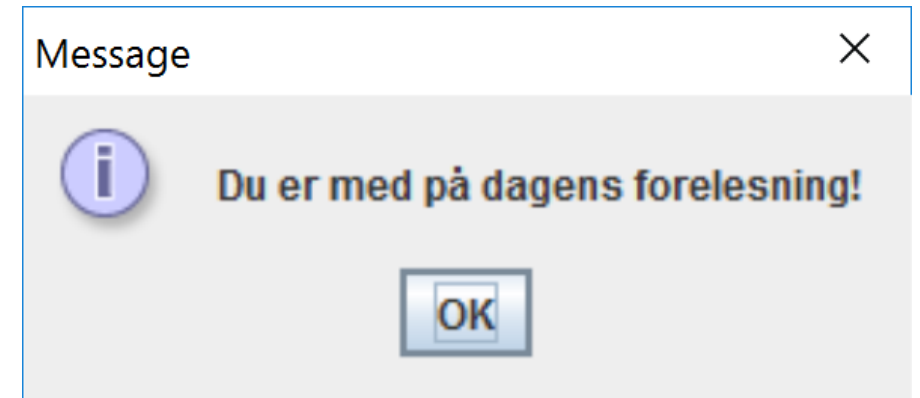
Fødselsdato (ddmmåå)

NesteAndre innloggingsmetoder

# JOptionPane

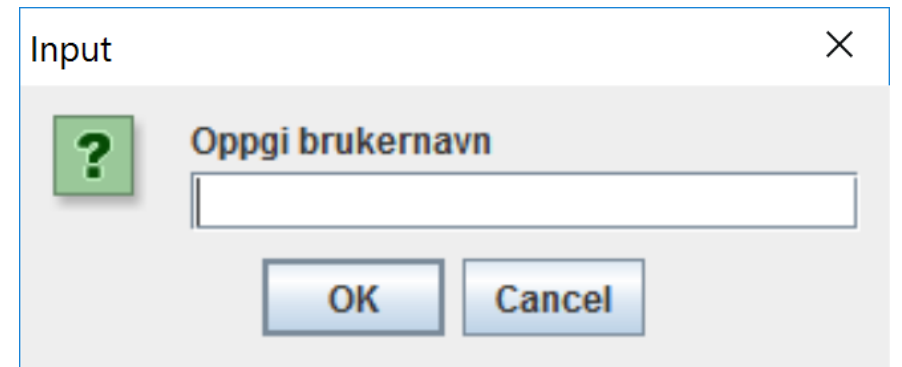
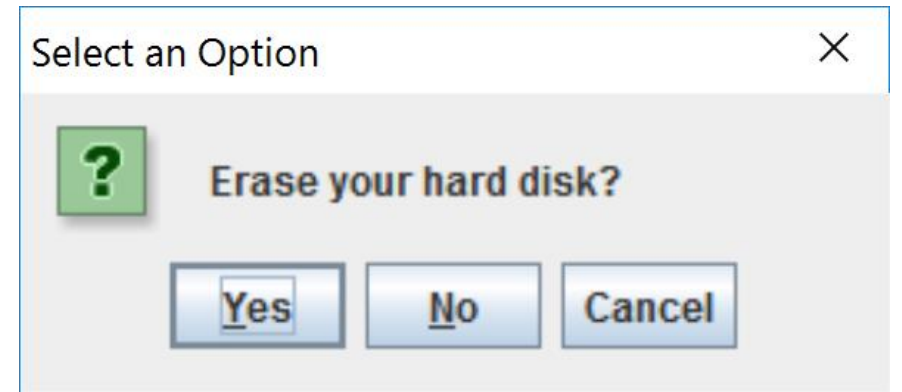
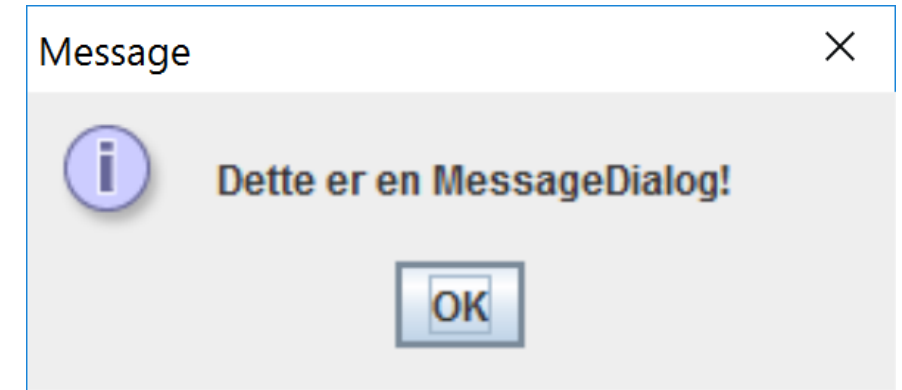
*En option pane er en enkel dialogboks for grafisk input/output.*

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



# JOptionPane

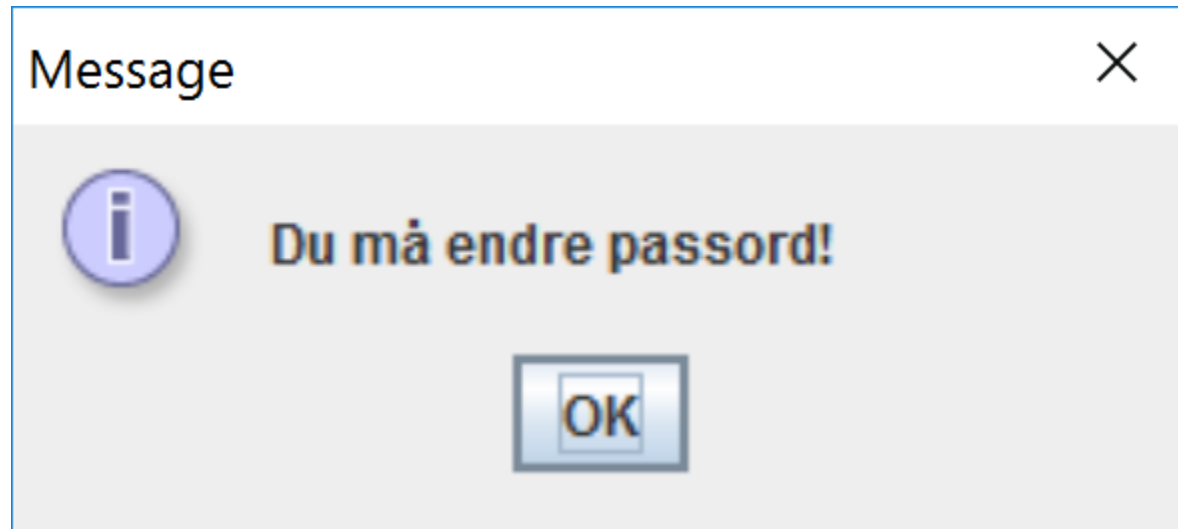
- `showMessageDialog(<parent>, <message>)`  
Viser en melding med en OK-knapp.
- `showConfirmDialog(<parent>, <message>)`  
Viser en melding og en liste med valg:  
Yes, No, Cancel;  
Returnerer brukerens valg som en `int` med en av følgende verdier:
  - `JOptionPane.YES_OPTION`
  - `JOptionPane.NO_OPTION`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

# showMessageDialog

Er en enkel dialogboks for output:



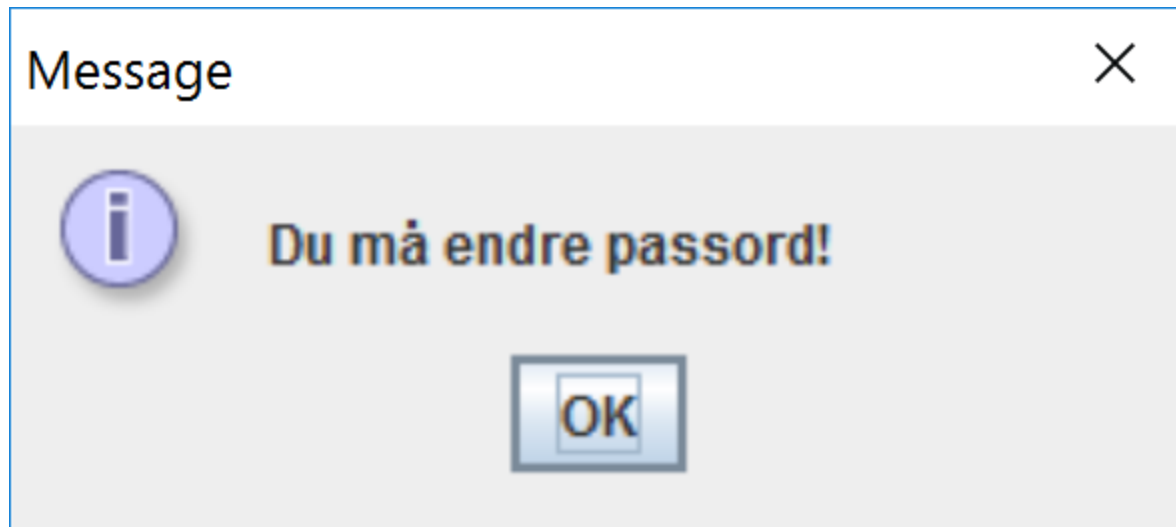
Tilsvare:

```
System.out.println("Du må endre passord!");
```

# showMessageDialog

```
showMessageDialog(  
    null, "Du må endre passord!");
```

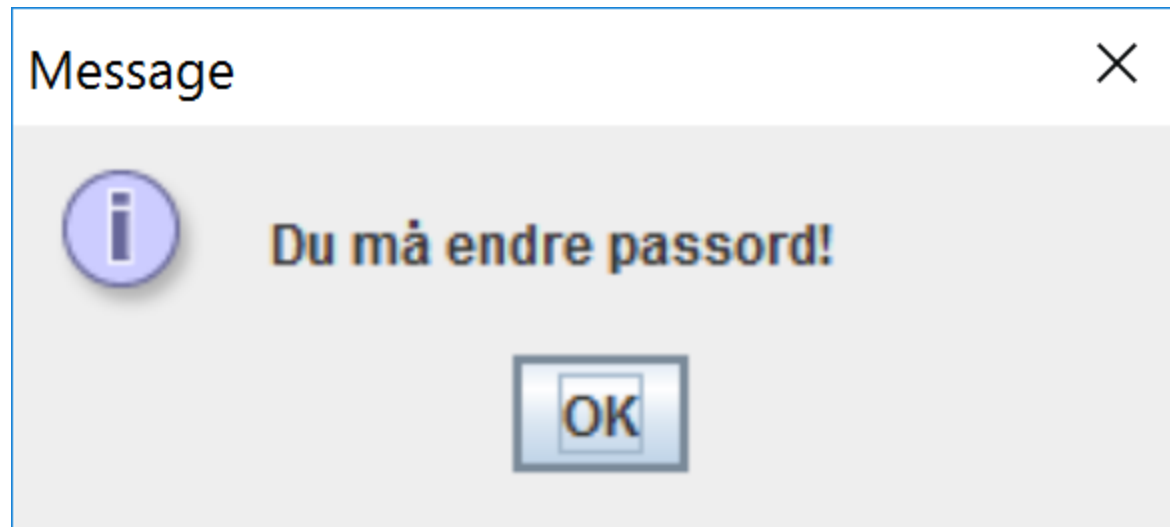
Midt på  
skjermen



```
import static javax.swing.JOptionPane.*;
```

# JOptionPane.showMessageDialog

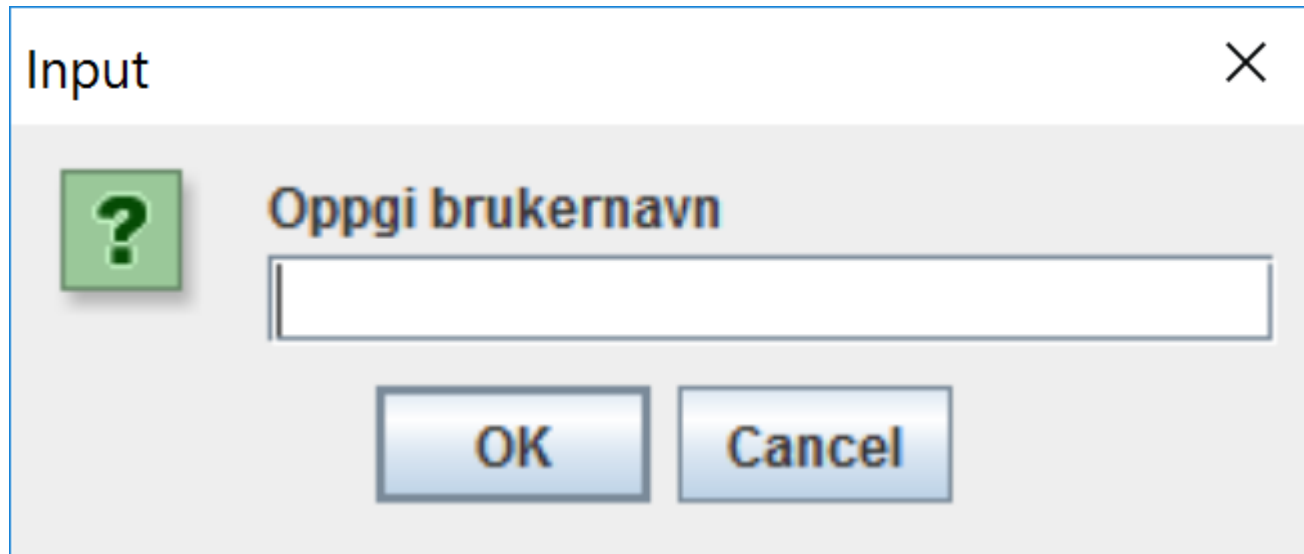
```
JOptionPane.showMessageDialog(  
    null, "Du må endre passord!");
```



```
import javax.swing.JOptionPane;
```

# showInputDialog

Er en enkel dialogboks for input:

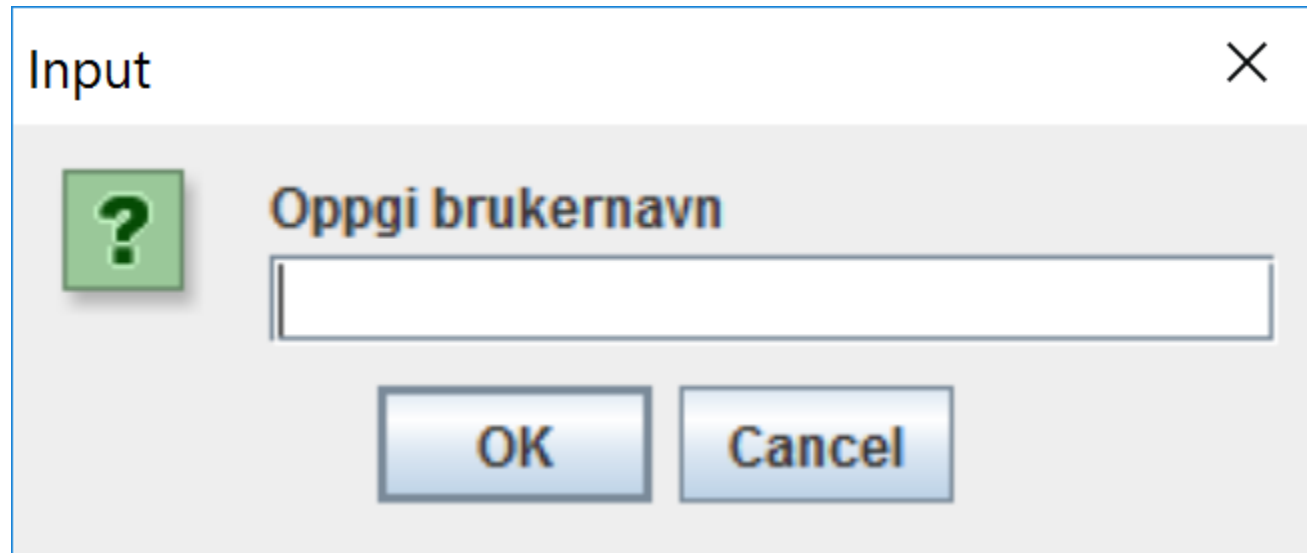


Tilsvare:

```
Scanner in = new Scanner(System.in);  
System.out.print("Oppgi brukernavn");  
String name = in.nextLine();  
(der in er en Scanner)
```

# showInputDialog

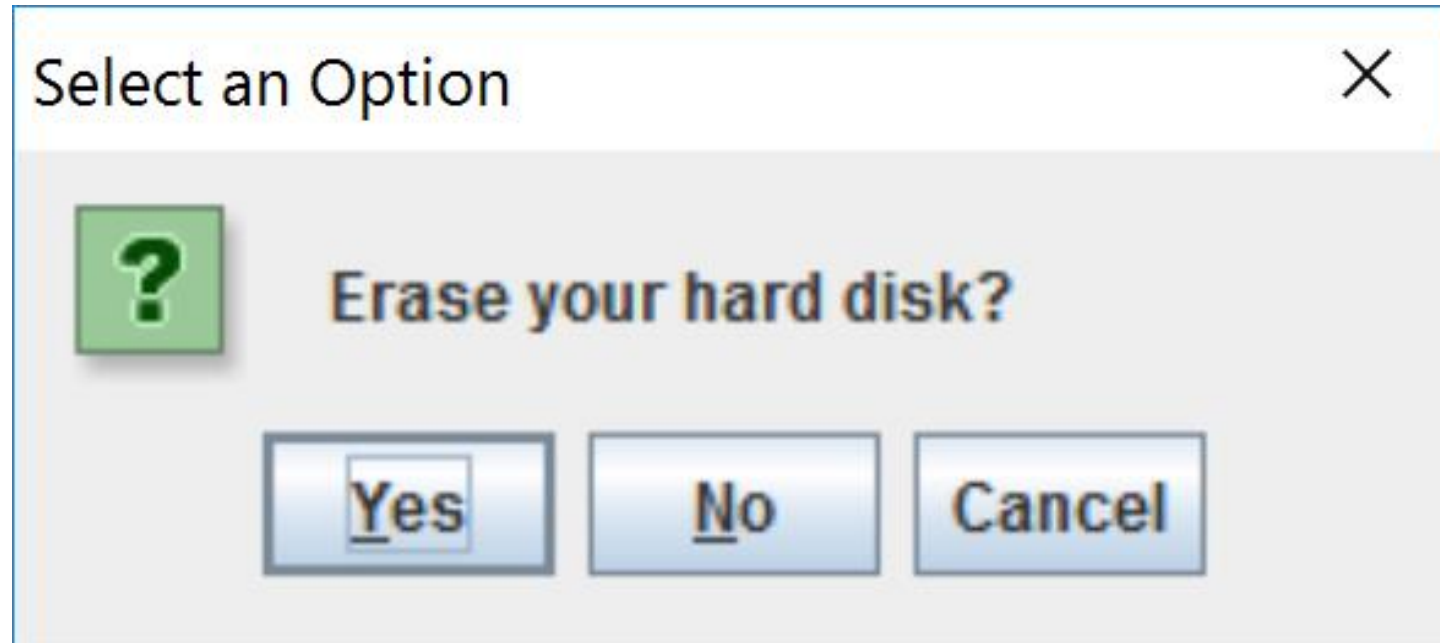
```
String name = showInputDialog(null,  
                                "Oppgi brukernavn");
```





# showConfirmDialog

Er en enkel dialogboks for å gjøre et valg:



Tilsvare:

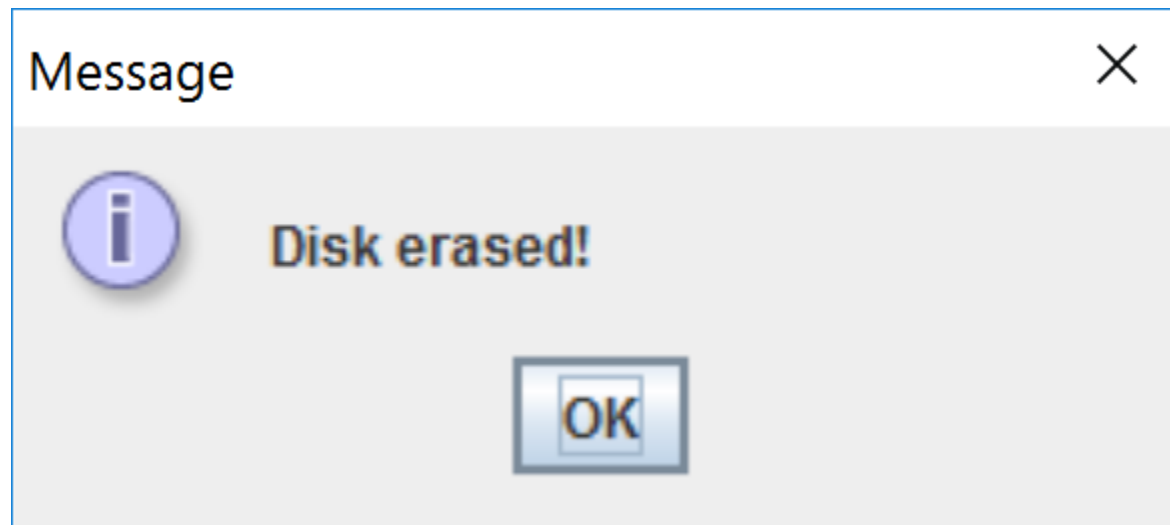
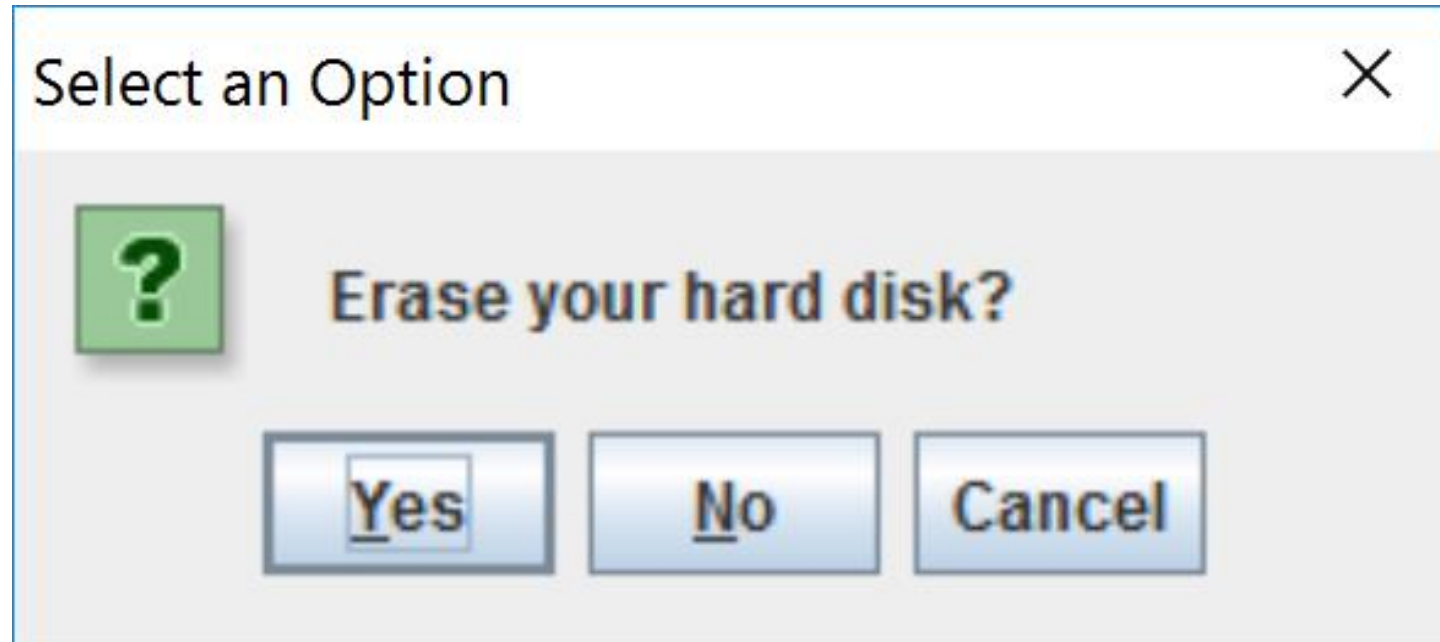
```
System.out.print("Erase your hard disk?");  
int choice = in.nextInt();
```

(der in er en Scanner)

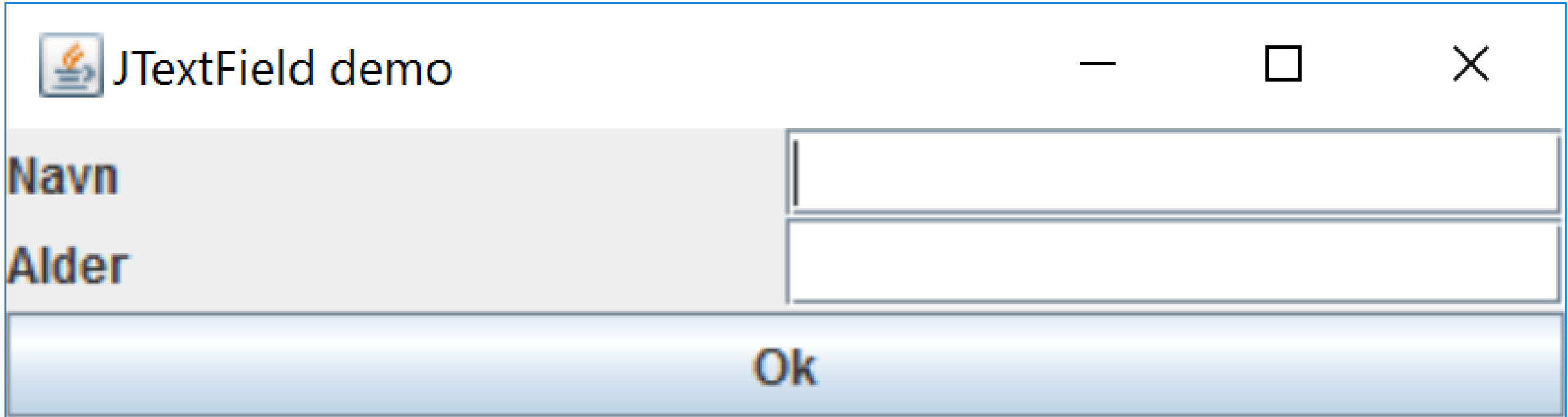
# showConfirmDialog

```
int choice = showConfirmDialog(  
    null, "Erase your hard disk?");  
if (choice == YES_OPTION) {  
    showMessageDialog(null, "Disk erased!");  
} else  
if (choice == NO_OPTION) {  
    showMessageDialog(null, "Disk not erased!");  
} else {  
    showMessageDialog(null, "Cancelled!");  
}
```

# showConfirmDialog



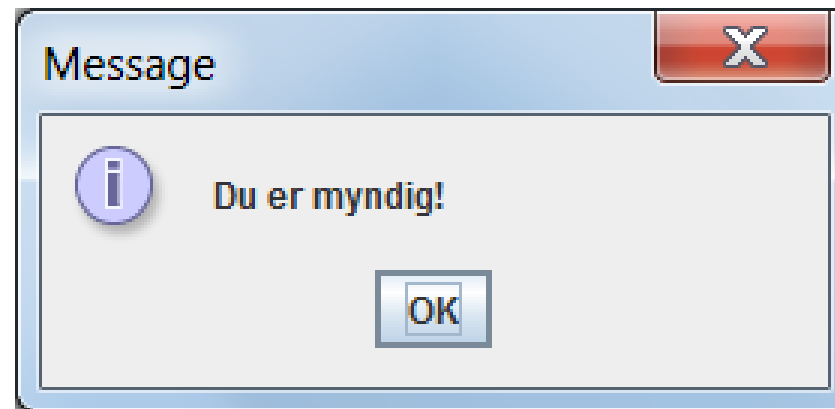
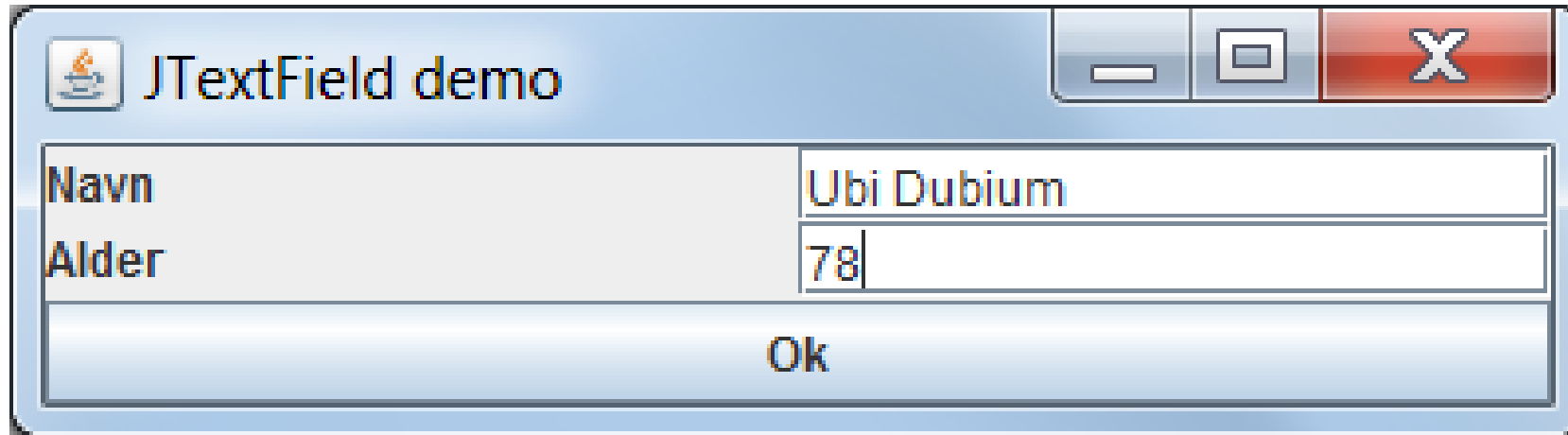
# JLabel/JTextField



The screenshot shows a Java Swing window titled "JTextField demo". The window has a standard title bar with a minimize button, a maximize button, and a close button. Inside the window, there is a light gray panel on the left containing two labels: "Navn" and "Alder". To the right of these labels are two empty text input fields (JTextField). Below the input fields is a large, light blue button with the text "Ok".

Når bruker skriver navn og alder, skal applikasjonen finne ut om personen er myndig eller ei.

# JLabel/JTextField



# Class Integer

Integer.parseInt

```
java.lang.Object  
    java.lang.Number  
        java.lang.Integer
```

## All Implemented Interfaces:

```
Serializable, Comparable<Integer>
```

---

```
public final class Integer  
    extends Number  
    implements Comparable<Integer>
```

The `Integer` class wraps a value of the primitive type `int` in an object. An object of type `Integer` contains a single field whose type is `int`.

In addition, this class provides several methods for converting an `int` to a `String` and a `String to an int` as well as other constants and methods useful when dealing with an `int`.

```
static int
```

```
parseInt(String s)
```

Parses the string argument as a signed decimal  
integer.

# JLabel/JTextField

```
import javax.swing.*;           //Layout
import java.awt.event.*;        //Action handling
import java.awt.*;              //GUI-komponenter
import static javax.swing.JOptionPane.*;

//Meldingsbokser
```



# JLabel/JTextField

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

```
public class JTextFieldExample extends JFrame  
                                implements ActionListener {
```

```
    private JTextField txtName;  
    private JTextField txtAlder;
```

//Hvorfor som fields?

```
    public JTextFieldExample() {
```

```
        super("JTextField demo");
```

//Alternativ til setTitle(...)

```
        JPanel pnlCenter = new JPanel(new GridLayout(2, 2));
```

# JLabel/JTextField

```
pnlCenter.add(new JLabel("Navn"));  
txtName = new JTextField();  
pnlCenter.add(txtName);  
pnlCenter.add(new JLabel("Alder"));  
txtAlder = new JTextField();  
pnlCenter.add(txtAlder);
```

# JLabel/JTextField

```
add(pnlCenter, BorderLayout.CENTER);
```

//Legger ut i vinduet

```
JButton btnOk = new JButton("Ok");
```

```
btnOk.addActionListener(this);
```

```
add(btnOk, BorderLayout.SOUTH);
```

//Legger ut i vinduet

# JLabel/JTextField

```
setSize(400, 110);  
setVisible(true);  
setDefaultCloseOperation(  
    JFrame.EXIT_ON_CLOSE);
```

```
}
```

# JLabel/JTextField

```
public void actionPerformed(ActionEvent event) {  
    String name = txtName.getText();  
    String strAlder = txtAlder.getText();  
    int alder = Integer.parseInt(strAlder);  
    if (alder >= 18) {  
        showMessageDialog(this, "Du er myndig!");  
    } else {  
        showMessageDialog(this, "Du er ikke myndig!");  
    }  
}
```

# Enkel kalkulator

Feltene Tall 1 og Tall 2 er tekstfelt

Tall 1	<input type="text"/>	txtNumber1
Tall 2	<input type="text"/>	

```
TextField txtNumber1
```

# Enkel kalkulator

Feltene `Tall 1` og `Tall 2` er tekstfelt

Det som skrives inn der må  
"hentes" i en `String`  
konverteres til et desimaltall.



# Class Double

```
java.lang.Object  
    java.lang.Number  
        java.lang.Double
```

## All Implemented Interfaces:

Serializable, Comparable<Double>

---

```
public final class Double  
    extends Number  
    implements Comparable<Double>
```

The `Double` class wraps a value of the primitive type `double` in an object. An object of type `Double` contains a single field whose type is `double`.

In addition, this class provides several methods for converting a `double` to a `String` and a `String` to a `double`, as well as other constants and methods useful when dealing with a `double`.



---

`static double`      **`parseDouble(String s)`**

Returns a new double initialized to  
the value represented by the  
specified `String`

---

# Enkel kalkulator

```
TextField txtNumber1
```

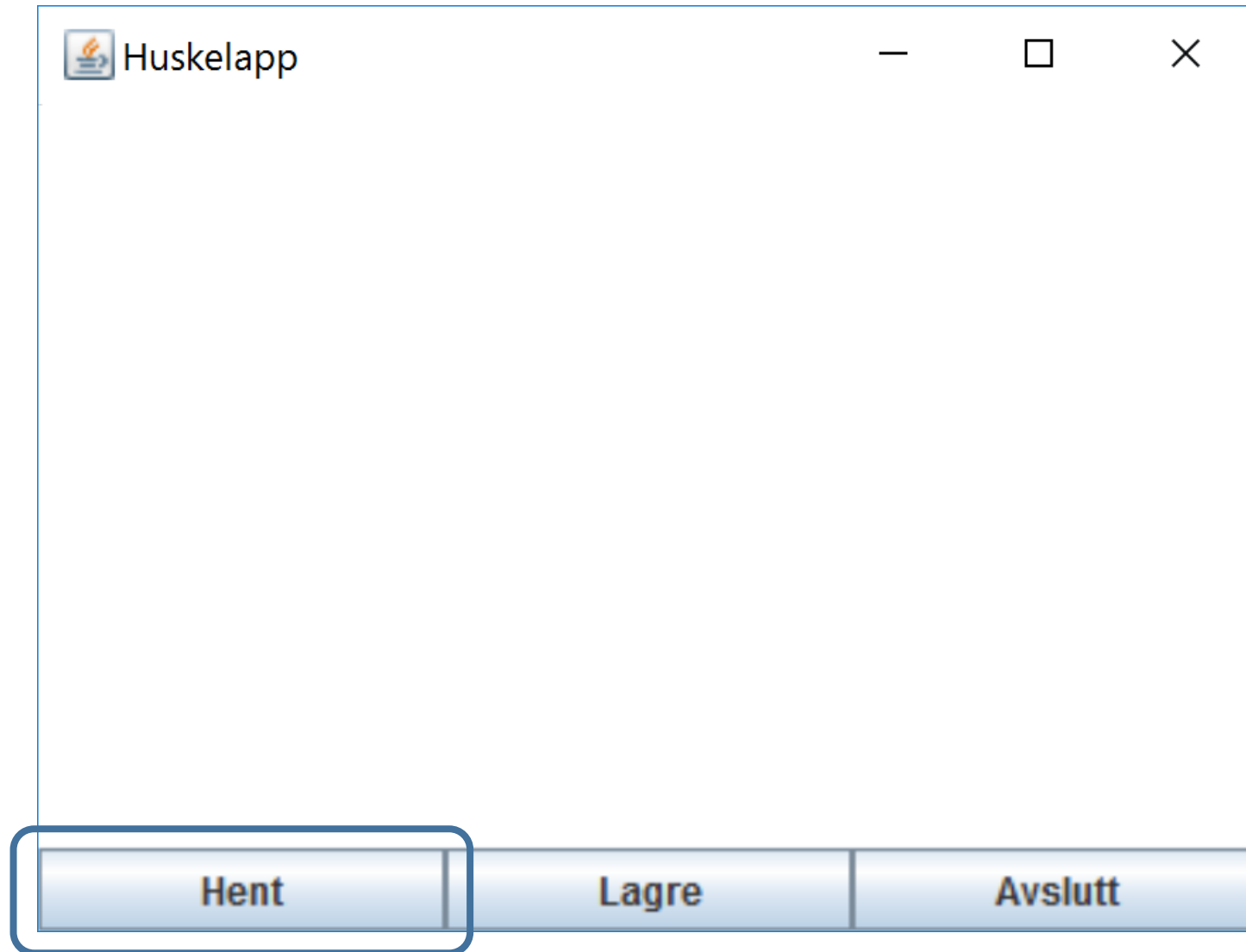
Hente data fra et tekstfelt:

```
String strNumber1 = txtNumber1.getText();
```

Konvertere tekst til et desimaltall:

```
double number1 = Double.parseDouble(strNumber1);
```

# Huskelapp-applikasjonen



# Huskelapp-applikasjonen

```
import java.awt.*;           //Layout
import java.awt.event.*;     //Action handling
import javax.swing.*;        //GUI-komponenter
import static javax.swing.JOptionPane.*; //Meldingsbokser
import java.io.*;            //Input/output
import java.util.Scanner;    //For lesing av fil
```

```
public class Note extends JFrame
    implements ActionListener {

    private JButton btnOpen;
    private JButton btnSave;
    private JButton btnExit;
    private JTextArea txaTextArea;
```

# Unntaks-håndtering

Når vi jobber med filer, må det lages et objekt av klassen File.

Ved behandling av filer, kan ting gå galt.

Når vi åpner eller lagrer en fil, kan det skje uventede ting.

Dette har JAVA-utviklerne tatt høyde for ved at det i slike situasjoner blir *kastet unntak*.

Dette er objekter som beskriver hva som har gått galt.

Og dette er objekter som programmet må være forberedt på å ta seg av.

Dette kalles *unntaks-håndtering*.

# Huskelapp-applikasjonen

Når det klikkes Hent, skal følgende ting skje:

1. Det skal dukke opp en melding om å oppgi filnavn
2. File-objektet må opprettes
3. Kobling mot fil opprettes
4. Fil-innhold "leses" og vises

# File

**File(String pathname)**

Creates a new `File` instance by converting the given `pathname` string into an abstract `pathname`.

# Scanner

```
Scanner sc = new Scanner(System.in);
```

```
int i = sc.nextInt();
```

//sc er koblet mot tastaturet

```
Scanner sc = new Scanner(new File("myNumbers"));
```

//sc er koblet mot en fil

```
long aLong = sc.nextLong();
```

//leser neste heltall fra fil



# PrintWriter

En PrintWriter kan brukes til å skrive til en fil:

---

**PrintWriter(File file)**

Creates a new PrintWriter, without automatic line flushing, with the specified file.

**void**

**println(String x)**

Prints a String and then terminates the line.

# Huskelapp-applikasjonen: actionPerformed

```
public void actionPerformed(ActionEvent e) {  
    Object clicked = e.getSource(); //Knappen som ble klikket  
    if (clicked == btnOpen) { //Det var Open-knappen  
        txtaTextArea.setText(""); //Tekstområdet "tømmes"  
    }  
}
```

```

try{
    String fileName =
        showInputDialog(this, "Filnavn");
    Scanner fileIn =
        new Scanner(new File(fileName));
    while (fileIn.hasNext()){
        String data = fileIn.nextLine();
        txtTextArea.append(data + "\n");
    }
    fileIn.close();
}catch (IOException ioex){
    JOptionPane.showMessageDialog(
        Note.this, "IO-feil: "
            + ioex.getMessage());
}

```

```

} else

```

//Filnavnet

//Lager en Scanner  
//Mer å lese?

//Leser en linje  
//Viser linjen

//Lukker filen

//Noe gikk feil...

```
} else
if (clicked == btnSave) {
    try{
        String fileName = //Filnavnet
            showInputDialog(this, "Filnavn");
        PrintWriter fileOut = //Lager en PrintWriter
            new PrintWriter(new File(fileName));
        fileOut.println(txaTextArea.getText());
        fileOut.close(); //Lagrer og lukker
    }catch (IOException ioex){
        JOptionPane.showMessageDialog(
            Note.this, "IO-feil: " //Noe gikk feil...
            + ioex.getMessage());
    }
}
```

# Huskelapp-applikasjonen: actionPerformed

```
} else {  
    System.exit(0);  
}  
  
}
```

```
public Note () {
```

```
    super("Huskelapp");
```

```
    JPanel pnlSouth =
```

```
        new JPanel(new GridLayout(1, 3));
```

```
    btnOpen = new JButton("Hent");
```

```
    btnSave = new JButton("Lagre");
```

```
    btnExit = new JButton("Avslutt");
```

```
    btnOpen.addActionListener(this);
```

```
    btnSave.addActionListener(this);
```

```
    btnExit.addActionListener(this);
```

```
    pnlSouth.add(btnOpen);
```

```
    pnlSouth.add(btnSave);
```

```
    pnlSouth.add(btnExit);
```

```
    txaTextArea = new JTextArea();
```

```
    add(txaTextArea, BorderLayout.CENTER);
```

```
    add(pnlSouth, BorderLayout.SOUTH);
```

```
    setSize(400, 300);
```

```
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
    setVisible(true);
```

```
}
```

# Huskelapp-applikasjonen

# Oppgaver

Se under Oppgaver 18.4.17