

---

# Klassen `javax.swing.JOptionPane`

Standardklassen `JOptionPane` er et alternativ til den hjemmelagede klassen `JavabokGUI`. API-referanser er samlet bakerst i dette notatet.

Til forskjell fra `JavabokGUI` skal vi nå bruke *klassemetoder* for å lese inn data fra brukeren. Dette er metoder som ikke er knyttet til noe bestemt objekt. For å få utført dem oppgir vi klassenavnet (og ikke et objektnavn, slik vi er vant med).

## Enkle innlesingsdialoger og meldingsbokser

Figur 1 viser kjøring av programmet i programliste 1.

---

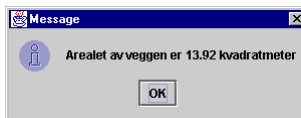
Figur 1 Kjøring av programmet i programliste 1



```
String lengdeLest =  
    JOptionPane.showInputDialog(  
        "Lengden av veggen (meter):");
```



```
String høydeLest =  
    JOptionPane.showInputDialog(  
        "Høyden av veggen (meter):");
```



```
JOptionPane.showMessageDialog(null,  
    "Arealet av veggen er " + arealet + " kvadratmeter");
```

---

Programliste 1

```
/*  
 * Veggberginger2.java E.L. 2001-05-07  
 */  
import javax.swing.JOptionPane;  
class Veggberginger2 {  
    public static void main(String[] args) {  
        String lengdeLest = JOptionPane.showInputDialog("Lengden av veggen (meter): ");  
        String høydeLest = JOptionPane.showInputDialog("Høyden av veggen (meter): ");  
        double lengde = Double.parseDouble(lengdeLest);  
        double høyde = Double.parseDouble(høydeLest);  
        double arealet = lengde * høyde;  
        JOptionPane.showMessageDialog(null,  
            "Arealet av veggen er " + arealet + " kvadratmeter");  
        System.exit(0);  
    }  
}  
/* Eksempeldata:
```

Lengde: 5.8 m

Høyde: 2.4 m

Arealet blir 13.92 kvadratmeter.

\*/

---

La oss se litt nærmere på dette programmet.

Klassemetoden `showInputDialog()` returnerer en referanse til en streng som inneholder det brukeren har skrevet inn. Vi bruker `parseDouble()` til å omforme fra streng til desimaltall. `parseDouble()` er en klassemetode i klassen `Double`, derfor står det `Double` foran metodenavnet. Vi har tilsvarende for heltall (`Integer.parseInt()`).

Metoden `showMessageDialog()` skriver en melding på skjermen.

Merk at metoden `showMessageDialog()` tar to argumenter, mens `showInputDialog()` tar ett. Det første argumentet i `showMessageDialog()` vil for alle våre formål inntil videre være `null`.

For at et program som bruker dialoger på denne måten skal stoppe på vanlig vis må vi til slutt legge inn setningen

```
System.exit(0);
```



---

Det ligger en del innebygd datakontroll i `JavabokGUI`. Denne kontrollen har du ikke her. Det vil si at:

- Du kan ikke lese data direkte inn til en tallvariabel. Du må lese inn til et streng-objekt og så omforme til tall.
- Dersom vi forventer at brukeren skriver inn et tall, og det som skrives inn ikke kan omformes til tall, kaster `parse`-metoden et unntaksobjekt (`ParseException`), og programmet stopper.
- Dersom vi lukker dialogvinduet, returnerer metodene en verdi (ofte `null`) som vi ikke bryr oss om å sjekke på i disse eksemplene. Derfor fører slik lukking av vinduet vanligvis også til unntak og programstopp.
- Ofte må du bruke musa for å klikke på knappene. Tastaturet fungerer bare delvis (sånn var det kanskje med `JavabokGUI` også?)

I slutten av dette notatet viser vi hvordan vi kan utføre datakontroll av denne typen.

---

Figur 2 Eksempler på meldingsdialoger



```
JOptionPane.showMessageDialog(null,  
"Den enkleste meldingsdialogen");
```



```
JOptionPane.showMessageDialog(null,  
"Viser feilmeldingssymbolet",  
"Metoden med fire parametre",  
JOptionPane.ERROR_MESSAGE);
```



```
ImageIcon bilde = new ImageIcon("blaa.gif");  
JOptionPane.showMessageDialog(null,  
"En gif-fil med blå firkant",  
"Metoden med fem parametre", 0, bilde);
```

Figur 2 viser flere eksempler på meldingsbokser. Det tredje eksemplet bruker et egetlaget bilde som ikon. Bildet ligger på en fil i samme katalog som programmet. Klassen `ImageIcon` ligger i pakken `javax.swing` og må importeres.

## Å stille ja/nei-spørsmål til brukeren

### Programliste 2

```
/*  
 * TestKalkulator1.java E.L. 2002-07-18  
 *  
 * Programmet leser inn to tall, lager et kalkulator-objekt (se programliste 5.1),  
 * og lar brukeren velge om han vil legge tallene sammen eller trekke fra.  
 * Regnestykket og resultatet skrives ut.  
 */  
  
import javax.swing.JOptionPane;  
class TestKalkulator1 {  
    public static void main(String[] args) {  
  
        /* Innlesing av data */  
        String tall1Lest = JOptionPane.showInputDialog("Første tall: ");  
        String tall2Lest = JOptionPane.showInputDialog("Andre tall: ");  
        double tall1 = Double.parseDouble(tall1Lest);  
        double tall2 = Double.parseDouble(tall2Lest);  
        int svar = JOptionPane.showConfirmDialog(null, "Skal tallene legges sammen? ",  
            "Kalkulator", JOptionPane.YES_NO_OPTION);  
  
        /* Beregning av resultater */  
        Kalkulator kalkus = new Kalkulator(tall1, tall2);
```

```
double beregnetSvar;
char operator;
if (svar == JOptionPane.YES_OPTION) { // Yes er trykket
    beregnetSvar = kalkus.beregnSum();
    operator = '+';
} else { // No eller Esc er trykket, eller dialogen er lukket
    beregnetSvar = kalkus.beregnDifferanse();
    operator = '-';
}

/* Utskrift av resultater */
String resultat = "Vi beregner: " + kalkus.finnTall1() + " " +
    operator + " " + kalkus.finnTall2();
resultat += "\nSvaret er " + beregnetSvar;
JOptionPane.showMessageDialog(null, resultat);
System.exit(0);
}
}

/*
Eksempel på kjøring, se figur 3
*/
```

---

Metoden for å stille ja/nei-spørsmål til brukeren har følgende hode:

```
public static int showConfirmDialog(Component forelder,
    Object melding, String tittel, int knappkombinasjon)
```

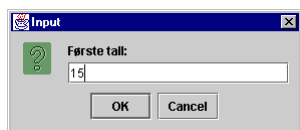
Knappekombinasjonen er en konstant i klassen `JOptionPane`. Vi bruker `YES_NO_OPTION` i programliste 2, API-referansen viser andre muligheter. Programmet bruker metoden på følgende måte:

```
int svar = JOptionPane.showConfirmDialog(null, "Skal tallene legges sammen? ",
    "Kalkulator", JOptionPane.YES_NO_OPTION);
```

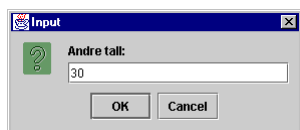
Avhengig av verdien til `svar` skal programmet legge sammen eller trekke fra. Vi bruker en `if`-setning:

```
if (svar == JOptionPane.YES_OPTION) {
    beregnetSvar = kalkus.beregnSum();
    operator = '+'; // vil bli del av resultatutskriften
} else {
    beregnetSvar = kalkus.beregnDifferanse();
    operator = '-';
}
```

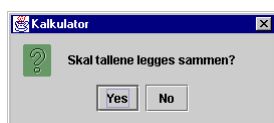
Figur 3 Kjøring av programmet TestKalkulator1



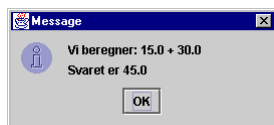
```
JOptionPane.showInputDialog("Første tall: ");
```



```
JOptionPane.showInputDialog("Andre tall: ");
```



```
JOptionPane.showConfirmDialog(null,  
"Skal tallene legges sammen? ",  
"Kalkulator", JOptionPane.YES_NO_OPTION);
```



```
JOptionPane.showMessageDialog(null, resultat);
```

## Å gi brukeren valget mellom flere alternativer

Programliste 3 viser et klientprogram der brukeren får valget mellom alle fire regneoperasjonene i klassen Kalkulator.

### Programliste 3

```
/*  
 * TestKalkulator2.java E.L. 2001-05-30  
 */  
import javax.swing.JOptionPane;  
class TestKalkulator2 {  
    public static void main(String[] args) {  
  
        /* Innlesing av data */  
        String tall1Lest = JOptionPane.showInputDialog("Første tall: ");  
        String tall2Lest = JOptionPane.showInputDialog("Andre tall: ");  
        double tall1 = Double.parseDouble(tall1Lest);  
        double tall2 = Double.parseDouble(tall2Lest);  
        String[] muligheter = {"pluss", "minus", "gange", "dele"};  
        int valg = JOptionPane.showOptionDialog(null, "Velg operator",  
        "Fire regneoperasjoner", 0, JOptionPane.PLAIN_MESSAGE,  
        null, muligheter, muligheter[0]);  
  
        /* Beregninger */
```

```
Kalkulator kalkus = new Kalkulator(tall1, tall2);
double svar = 0.0;
char operator = ' '; // Bruker denne i resultatutskriften
boolean ok = true;
if (valg == 0) {
    operator = '+';
    svar = kalkus.beregnSum();
} else if (valg == 1) {
    svar = kalkus.beregnDifferanse();
    operator = '-';
} else if (valg == 2) {
    svar = kalkus.beregnProdukt();
    operator = '*';
} else if (valg == 3) {
    operator = '/';
    if (tall2 == 0.0) ok = false; // må unngå divisjon med 0
    else svar = kalkus.beregnKvotient();
} else { // dialogen er lukket
    ok = false;
}

/* Utskrift av resultatet */
String resultat;
if (ok) resultat = tall1 + " " + operator + " " + tall2 + " = " + svar;
else resultat = "Kan ikke beregne resultat.";
JOptionPane.showMessageDialog(null, resultat);
System.exit(0);
}
}
/* Kjøre-eksempler:

Første tall: 12.5
Andre tall: 3.56
Velger operasjonen "gange"
Resultatutskrift: 12.5 * 3.56 = 44.5

Første tall: 12.5
Andre tall: 0.0
Velger operasjonen "dele"
Resultatutskrift: Kan ikke beregne resultat.
*/
```

---

Fire trykknapper (se figur 4) gir deg valget mellom de fire aritmetiske regneoperasjonene. Vi skal se hvordan vi bruker metoden `showOptionDialog()` til å vise disse mulighetene.

De to siste parametrene (se API-referansen bakerst i notatet) ser slik ut: `Object[] muligeValg`, `Object startvalg`.

---

Hva betyr datatypen `Object`? Fra figur 4.6 side 113 i læreboka ser vi at klassen `Object` er superklasse til alle andre klasser. Det betyr at den beskrivelsen som er gitt ved klassen `Object` gjelder alle mulige objekter. I praksis betyr dette at dersom parametertypen er `Object`, så kan argumentet tilhøre en hvilken som helst klasse. Vi lar argumentet tilhøre datatypen `String`.

Aktuelle valgalternativer setter vi altså opp slik:

```
String[] muligheter = {"pluss", "minus", "gange", "dele"};
```

Som argument til parameteren `startValg` skal vi sende inn den verdien vi ønsker at skal være valgt i det brukeren får opp listen. Her setter vi denne verdien lik den første verdien i tabellen, det skriver vi slik: `muligheter[0]`.

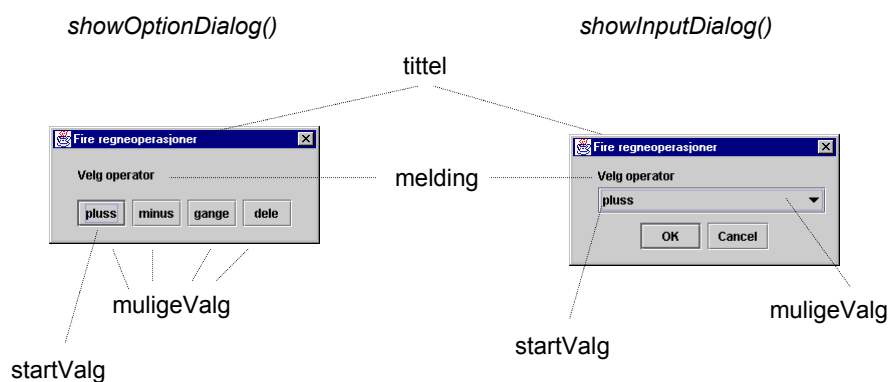
Kallet på metoden `showOptionDialog()` ser dermed slik ut:

```
int valg = JOptionPane.showOptionDialog(null, "Velg operator",  
    "Fire regneoperasjoner", 0, JOptionPane.PLAIN_MESSAGE,  
    null, muligheter, muligheter[0]);
```

Valget brukeren har gjort ligger nå lagret i variabelen `valg`. Her finner vi nummeret til det valgte elementet i tabellen `muligeValg`. Det vil si at `valg` lik 0 betyr "pluss", `valg` lik 1 betyr "minus", `valg` lik 2 betyr "gange", og `valg` lik 3 betyr "dele".

---

Figur 4 Eksempel på bruk av `showOptionDialog()` og `showInputDialog()`



---

Tabell 1 og venstre del av figur 4 illustrerer sammenhengen mellom parametre og argumenter i metoden `showOptionDialog()`. Høyre del av figuren viser dialogboksen dersom vi i stedet bruker `showInputDialog()`. Kallet i programmet vil i tilfelle se slik ut:

```
Object valg = JOptionPane.showInputDialog(null, "Velg operator",  
    "Fire regneoperasjoner", JOptionPane.DEFAULT_OPTION, null,  
    muligheter, muligheter[0]);
```

Returtypen fra metoden er `Object`, og vi må sammenligne returverdien med tabellelementene:

```
if (valg == muligheter[0]) { // "pluss" er trykket1
```

---

Tabell 1 Sammenheng mellom parametre og argumenter i metoden `showOptionDialog()`

parameter	argument (eksempel)	forklaring
<code>Component forelder</code>	<code>null</code>	Argument lik <code>null</code> betyr at dialogen ikke er en del av et større system med vinduer, menyer, etc.
<code>Object melding</code>	<code>"Velg operator"</code>	Se figur 4.
<code>String tittel</code>	<code>"Fire regneoperasjoner"</code>	Se figur 4.
<code>int knappe-kombinasjon</code>	<code>0</code>	Knappene som skal vises er bestemt av <code>muligeValg</code> . Så lenge <code>muligeValg</code> er forskjellig fra <code>null</code> , er verdien til denne parameteren uten betydning.
<code>int meldingstype</code>	<code>JOptionPane.PLAIN_MESSAGE</code>	Her angis eventuelt standardikon som skal vises til venstre i dialogen. I eksemplet vises intet ikon. Et aktuelt alternativ til <code>PLAIN_MESSAGE</code> er <code>QUESTION_MESSAGE</code> .
<code>Icon ikon</code>	<code>null</code>	Dersom du ønsker ditt eget ikon i stedet for et av standardikonene, kan du sette opp det her. Eksempel, se figur 2.
<code>Object[] muligeValg</code>	<code>muligheter</code>	Her skal du sende inn en tabell som inneholder valgmulighetene. Det er mest aktuelt med en tabell av <code>String</code> eller en tabell av <code>ImageIcon</code> .
<code>Object startValg</code>	<code>muligheter[0]</code>	Når dialogboksen kommer opp på skjermen skal en av knappene være merket som valgt, her vil det bli første alternativ i tabellen <code>muligheter</code> .

---

1. Her sammenligner vi referansene, og ikke innholdet i strengene. Idette tilfellet er det korrekt, men resultatet ville selvfølgelig blitt det samme om vi hadde brukt `equals()` (se side 148 i boka).

---

## ***Kontroll av inndata***

Med utgangspunkt i metodene i `JOptionPane` kan vi relativt enkelt lage innlesingsmetoder med datakontroll.

Programliste 4 viser klassemetoder for innlesing av tekst, desimaltall og heltall. Prøvekjør programmet. Kontroll av data er lagt inn etter følgende spesifikasjoner:

- Dersom brukeren trykker Cancel, lukker vinduet eller ikke skriver inn noe data, kommer meldingen “Du må oppgi data!”.
- Dersom tall skal skrives inn, og brukeren skriver inn en tekst som ikke kan tolkes som et tall, kommer meldingen “Ugyldig heltall!” eller “Ugyldig desimaltall!”.

Metodene spør på nytt inntil brukeren skriver inn gyldige data.

Vi skal se litt nærmere på de enkelte metodene. Vi begynner med metoden `lesTekst()`. Vi bruker `showInputDialog()` til å lese inn data. Denne metoden returnerer `null` dersom brukeren trykker Cancel eller lukker vinduet. Følgende kodebit kontrollerer dette og skriver ut meldingen “Du må oppgi data!” før brukeren blir spurt på nytt:

```
String tekst = JOptionPane.showInputDialog(ledetekst);
while (tekst == null || tekst.trim().equals("")) { // tillater heller ikke tomt felt
    JOptionPane.showMessageDialog(null, "Du må oppgi data!");
    tekst = JOptionPane.showInputDialog(ledetekst);
}
```

Legg merke til hvordan løkken er bygget opp. Programkontrollen går bare inn i løkken dersom teksten ikke er gyldig.

Vi kan bruke `do-while`-setningen dersom vi ikke skal skrive ut en melding om feilen:

```
String tekst;
do {
    tekst = JOptionPane.showInputDialog(ledetekst);
} while (tekst == null || tekst.trim().equals(""));
```

Fall ikke for fristelsen til å bruke `do-while` dersom melding skal skrives ut! Du må i tilfelle ha en `if`-setning etter innlesingen inne i løkkroppen, og det blir lite elegant.

Metodene for å lese inn tall er vanskeligere. Som kjent bruker vi `parseInt()` eller `parseDouble()` til å omforme fra tekst til tall. Dersom teksten ikke kan omformes til et gyldig tall, har vi hittil fått en feilmelding i konsollvinduet, og programmet har stoppet. Dette er selvfølgelig en uheldig situasjon, og vi skal nå vise hvordan vi kan fange opp dette tilfellet og også nå spørre brukeren på nytt.

---

Til å lese inn teksten bruker vi metoden `lesTekst()` gjennomgått foran. Denne metoden kan vi kalle direkte uten kvalifisering, ettersom vi er i samme klasse.

```
String lestTekst = lesTekst(ledetekst); // kaller en annen metode i samme klasse
```

Tolkning av teksten og håndtering av eventuell feil som inntreffer får vi til på følgende måte:

```
try { // vi skal prøve å omforme teksten til tall
    tall = Integer.parseInt(lestTekst); // her prøver vi
    .....hit kommer vi bare dersom det gikk bra
} catch (NumberFormatException e) { // hit kommer vi bare dersom det ikke gikk bra
    JOptionPane.showMessageDialog(null, "Ugyldig desimaltall!\n");
}
```

Her er det mye ukjent.

Dersom metoden `parseInt()` ikke klarer å omforme teksten til et gyldig tall, sier vi at den “kaster et unntaksobjekt”. Vi ser ikke at objektet kastes (inne i metoden `parseInt()` finnes det et sted faktisk en `throw`-setning), men vi må ta i mot det dersom det blir kastet, ellers kommer den stygge feilmeldingen nevnt foran.

Kallet på `parseInt()` ligger inne i en blokk som starter med det reserverte ordet `try`. Det signaliserer at vi skal prøve noe vi ikke er sikker på utfallet av. Dersom det ikke går bra vil et unntaksobjekt bli kastet. Dette skal vi ta i mot. Og det skjer i en `catch`-blokk litt lenger ned i programkoden.

Studer kommentarene i koden over. Merk deg hvilke setninger som blir utført når feil oppstår, og hvilke som blir utført dersom alt går bra.

I metoden `lesHeltall()` i programliste 4 bruker vi en logisk variabel, `ok`, til å holde orden på hvorvidt omformingen gikk bra eller ikke. Vi setter variabelen lik `false` i begynnelsen av metoden, og den settes til `true` bare dersom vi kommer forbi kallet på `parseInt()` uten at feil har inntruffet.

Fortvil ikke dersom du synes dette er vanskelig.<sup>1</sup> Kopier klassen og bruk metodene der du trenger dem. Husk å kvalifisere med klassenavnet:

```
String tekst = LesData.lesTekst("Skriv en tekst: ");
```

#### Programliste 4

---

```
/*
 * TestLesData.java E.L. 2000-05-31
 */
import javax.swing.JOptionPane;
class LesData {

    public static String lesTekst(String ledetekst) {
```

---

1. Unntakshåndtering er tema i kapittel 8, og vi kommer da tilbake til flere detaljer omkring dette.

---

```

String tekst = JOptionPane.showInputDialog(ledetekst);
while (tekst == null || tekst.equals("")) {
    JOptionPane.showMessageDialog(null, "Du må oppgi data!");
    tekst = JOptionPane.showInputDialog(ledetekst);
}
return tekst;
}

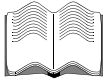
public static int lesHeltall(String ledetekst) {
    int tall = 0;
    boolean ok = false;
    do {
        String lestTekst = lesTekst(ledetekst);
        try {
            tall = Integer.parseInt(lestTekst);
            ok = true;
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Ugyldig heltall!\n");
        }
    } while (!ok);
    return tall;
}

public static double lesDesimaltall(String ledetekst) {
    double tall = 0;
    boolean ok = false;
    do {
        String lestTekst = lesTekst(ledetekst);
        try {
            tall = Double.parseDouble(lestTekst);
            ok = true;
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Ugyldig desimaltall!\n");
        }
    } while (!ok);
    return tall;
}
}

class TestLesData {
    public static void main(String[] args) {
        String tekst = LesData.lesTekst("Skriv en tekst: ");
        double desimaltall = LesData.lesDesimaltall("Skriv et desimaltall: ");
        int heltall = LesData.lesHeltall("Skriv et heltall: ");
        System.out.println("Du skrev dette: ");
        System.out.println(tekst);
        System.out.println(desimaltall);
        System.out.println(heltall);
        System.exit(0);
    }
}

```

---



## API-referanse

---

### *Klassen `javax.swing.JOptionPane`*

Metodene har flere felles parametre:

- `forelder` vil være null så lenge dialogboksen ikke tilhører et ordinært vindu i et grafisk brukergrensesnitt.
- `melding` er teksten som skrives i dialogen. Selv om parametertypen er `Object`, så går det bra med `String` som argument.<sup>1</sup>
- `tittel` er teksten som kommer øverst i tittellinjen i dialogen.
- `meldingstype` bestemmer ikonet til venstre i boksen og kan være én av følgende klassekonstanter: `ERROR_MESSAGE`, `INFORMATION_MESSAGE`, `WARNING_MESSAGE`, `QUESTION_MESSAGE` eller `PLAIN_MESSAGE` (intet ikon).

Dersom vi bruker en av metodene der `meldingstype` ikke skal oppgis, får vi fram et standardikon.

- `ikon` representerer et bilde. Parametertypen er `Icon`, argumentet kan være av typen `ImageIcon`,<sup>2</sup> som er velegnet når vi har bildet på en fil. Dette argumentet overstyrer argumentet til `meldingstype`.
- knappekombinasjon kan være en av følgende klassekonstanter i klassen `JOptionPane`: `DEFAULT_OPTION`, `YES_NO_OPTION`, `YES_NO_CANCEL_OPTION` eller `OK_CANCEL_OPTION`. Den enkleste utgaven av metoden krever ikke at knappekombinasjon oppgis, da gjelder `YES_NO_CANCEL_OPTION`.

#### *Meldingsdialoger:*

```
public static void showMessageDialog(Component forelder, Object melding)
public static void showMessageDialog(Component forelder, Object melding,
                                   String tittel, int meldingstype)
public static void showMessageDialog(Component forelder, Object melding,
                                   String tittel, int meldingstype, Icon ikon)
```

Dialogene viser en melding med en OK-knapp.

- 
1. I praksis kan vi sende inn et objekt av en hvilken som helst klasse. Objektet kan være en GUI-komponent (kapittel 13), eller det kan tilhøre klassen `ImageIcon`. I andre tilfeller blir objektets `toString()`-metode brukt (se kapittel 10).
  2. `Icon` er et interface (se kapittel 10). Argumentet kan være av en hvilken som helst klasse som implementerer dette interfacet.

---

### *Inputdialoger:*

```
public static String showInputDialog(Component forelder, Object melding)
public static String showInputDialog(Component forelder, Object melding,
                                   String tittel, int meldingstype)
public static String showInputDialog(Object melding)
```

Metodene lar brukeren skrive inn en tekst som returneres til klienten.

Metodene returnerer null dersom brukeren trykker Cancel-knappen eller lukker dialogen ved å trykke i øverste høyre hjørne.

### *Bekreftelsesdialoger:*

```
public static int showConfirmDialog(Component forelder, Object melding)
public static int showConfirmDialog(Component forelder, Object melding,
                                   String tittel, int knappkombinasjon)
public static int showConfirmDialog(Component forelder, Object melding,
                                   String tittel, int knappkombinasjon, int meldingstype)
public static int showConfirmDialog(Component forelder, Object melding,
                                   String tittel, int knappkombinasjon, int meldingstype, Icon ikon)
```

Dette er dialoger med forskjellige kombinasjoner av OK-, Yes-, No- og Cancel-knapper.

Metoden returnerer informasjon om hvilken knapp brukeren trykket på. Aktuelle alternativer er gitt ved klassekonstantene (klassen `JOptionPane`) `CANCEL_OPTION`, `OK_OPTION`, `YES_OPTION`, `NO_OPTION`, eller `CLOSED_OPTION` hvis brukeren lukket dialogen.

### *Valgdialoger:*

```
public static int showOptionDialog(Component forelder, Object melding,
                                   String tittel, int knappkombinasjon, int meldingstype, Icon ikon,
                                   Object[] muligeValg, Object startValg)
public static Object showInputDialog(Component forelder, Object melding,
                                   String tittel, int meldingstype, Icon ikon, Object[] muligeValg,
                                   Object startValg)
```

Metodene gjør det mulig for brukeren å velge blant flere alternativer.

`showOptionDialog()` viser valgene som trykknapper, mens `showInputDialog()` viser valgene i form av en liste (nedtrekksliste dersom færre enn 20 alternativer).

`muligeValg` er en tabell med valgmuligheter. Det kan være tekster eller ikoner. `startValg` viser hva som skal være valgt dersom bruker ikke foretar eget valg. `showOptionDialog()` returnerer nummeret til det valget som er gjort. Første valg har nummer 0. Dersom brukeren trykker Cancel eller lukker dialogen returneres konstanten `CLOSED_OPTION`.

`showInputDialog()` returnerer en referanse til det valgte objektet blant de referansene som er sendt inn i tabellen `muligeValg`. Dersom brukeren trykker Cancel, returneres null.