Classe JDialog i JOptionPane

JDialog és un Component que serveix per **presentar diàlegs** que són **finestres auxiliars o secundàries**, serveixen per prevenció o, si no es pot utilitzar per donar informació sobre alguna cosa. Està entre JFrame, que seria la finestra principal i JOptionPane, que serien finestres secundàries per mostrar notificacions, demanar confirmacions o demanar alguna dada ràpida a l'usuari.
Un **JDialog** és més adequat per presentar una finestra secundària amb més contingut que un JOptionPane.

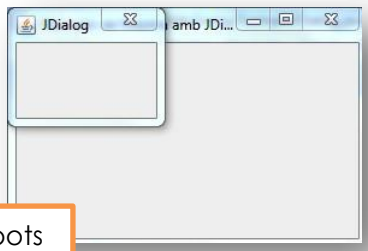
Els diàlegs que **JDialog** mostra poden **ser modals o no modals**, això vol dir que si són modals la finestra del diàleg bloqueja les entrades a altres finestres.

Tots els diàlegs depenen **d'un frame**, les modificacions que se li facen al **frame** afectaran el diàleg: en el cas que el frame siga tancat, minimitzat o maximitzat, els seus diàlegs tindran el mateix comportament.

Un JDialog sempre ha de tenir un **component *parent*** de el qual es derive, que ha de ser un contenidor **d'alt nivell i sobre el qual aplicarà el seu efecte modal o no**.

```
package exemplesSwing;

import javax.swing.*;
public class ExempleJDialog extends JFrame {

       // el constuctor
    public ExempleJDialog() {

       super("Exemple de finestra amb JDialog davant!");

       setSize(300,200);
       setVisible(true);

       JDialog cuadroDialogo= new JDialog(this, true); //Crea un dialog modal
      cuadroDialogo.setTitle("JDialog");
       cuadroDialogo.setSize(100,100);
      cuadroDialogo.setVisible(true);
    }

    public static void main(String[] args) {

       SwingUtilities.invokeLater(new Runnable() {
```

```
            @Override
            public void run() {
                new ExempleJDialog();
            }
        });

    }
}
```



Si executes el codi veuràs que **no** pots **accedir** a el botó de tancar **el JFrame** pare perquè el diàleg és **modal**.

Si ara canviem la línia

```
JDialog cuadroDialogo = new JDialog(this, true);
```

per

```
JDialog cuadroDialogo = new JDialog(this, false);
```

el convertim en NO Modal, i per tant podem accedir a la finestra pare i tancar-la.

Un altre exemple sobre finestres JDialog modal: De vegades les finestres de JOptionPane no són suficients pel que necessitem. Per exemple, podem voler demanar una dada més complexa que una simple selecció d'una opció. I de vegades mentre l'usuari introdueix aquesta dada complexa a la finestra i prem "Acceptar", volem que el nostre codi es pare esperant la resposta.

Per a aquestes situacions tenim els **JDialog modals**. En un JDialog podem posar tots els components que vulguem, fent la finestra tan complexa com vulguem. Si aquest JDialog és a més modal, en el moment de fer-la visible cridant a setVisible(true), el codi es quedarà parat en aquesta crida fins que la finestra es tanque. Podem, per tant, just després del setVisible(true) demanar a la finestra les dades que ha introduït l'operador.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ExempleDialogModal extends JDialog {
    private JTextField textField;

    /**
     * Constructor que pone titulo al dialogo, construye la ventana y
la hace
     * modal.
     *
     * @param padre
     *            Frame que hace de padre de esta dialogo.
     */
    public ExempleDialogModal(Frame padre) {
        // padre y modal
        super(padre, true);
        setTitle("Mete un dato");
        textField = new JTextField(20);
        getContentPane().add(textField);

        // Se oculta la ventana al pulsar <enter> sobre el textfield
        textField.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent arg0) {
                setVisible(false);
            }
        });
    }

    /**
     * Devuelve el texto en el jtextfield
     *
     * @return el texto introducido por el usuario después de que se
cierre la ventana.
     */
    public String getText() {
        return textField.getText();
    }

    public static void main(String[] args) {
        //Nos creamos la ventana padre
        JFrame ventanaPadre = new JFrame();
        ventanaPadre.setTitle("Ejemplo JDialog"); //Título del JFrame
        ventanaPadre.setSize(250,200); //Dimensiones del JFrame
        ventanaPadre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//Cerrar proceso al cerrar ventana
        ventanaPadre.setVisible(true); //Mostrar JFrame


        ExempleDialogModal dialogoModal = new
ExempleDialogModal((Frame) ventanaPadre);
        dialogoModal.pack();  // para darle tamaño automático a la
ventana.
        dialogoModal.setVisible(true);

    // Al ser modal, la llamada a setVisible(true) se queda bloqueada
hasta
    // que el dialogo modal se oculte. Por ello, a continuación
tenemos
    // la seguridad de que el texto ya esta disponible.
        System.out.println("El usuario ha escrito
"+dialogoModal.getText());
    }
}
```

JOptionPane – ShowMessageDialog i ShowInputDialog

Un JOptionPane és una finestra emergent d'una aplicació java swing que es fa servir per mostrar notificacions a l'usuari.

Un JOptionPane sol tindre un títol de finestra, un text amb el missatge que es vol mostrar a l'usuari, una icona que pot indicar-li la gravetat del missatge, opcionalment algun tipus de formulari senzill estil caixa de text o JComboBox i una sèrie de botons per tancar el JOptionPane, com a Ok, Acceptar, Cancel·lar, Sí, No.

Els JOptionPane són finestres modals. Això vol dir que fins que l'usuari no la tanque, no pot tocar les altres finestres de la interfície.

Quines opcions té un JOptionPane?

Les opcions que ofereix JOptionPane són:

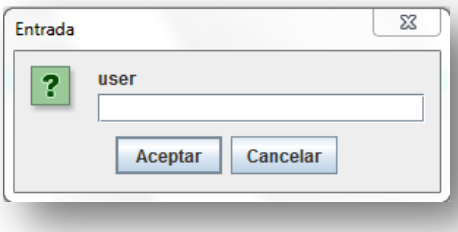
- Mostrar un missatge informatiu a l'usuari
- Demanar una confirmació a l'usuari abans de fer una operació
- Demanar una dada ràpida a l'usuari.

Si parlem de mètodes de JOptionPane, teniu el mètode general **showOptionDialog()** que val per a les tres opcions. Hi hem de donar tots els detalls de la finestra. És a dir, text, títol, icona, botons que volem, si cal demanar alguna dada a l'usuari, etc. Com que aquest mètode té molts paràmetres i és una mica molest d'usar, JOptionPane ens ofereix mètodes simplificats per a les tres opcions bàsiques.

- showMessageDialog() : Finestra emergent per mostrar informació a l'usuari. Pot ser simplement una informació, avís, error, etc.
- showConfirmDialog() : Finestra emergent per demanar confirmació a l'usuari amb opcions com SI, NO, CANCEL·LAR. Per exemple, si esteu segurs de voler esborrar alguna dada de base de dades.
- showInputDialog() : Finestra emergent per demanar a l'usuari alguna dada ràpida, com ara un text, una opció d'un menú, un valor sencer, etc.

Aquests són els mètodes per fer finestres emergents.

Vegem com funcionen les caixes de diàleg veient un exemple pràctic. Utilitzarem les caixes de diàleg **MessageDialog** que mostren informació i **InputDialog** que demanen informació i el programa recull aquesta informació:




```
package exemplesSwing;

import javax.swing.*;
import javax.swing.JOptionPane;


public class ExempleJDialogShow {

    public static void main(String[] args) {

       String user = JOptionPane.showInputDialog(null, "user",
```

```
        JOptionPane.QUESTION_MESSAGE);  // la icona serà un iterrogant
        );
         String password = JOptionPane.showInputDialog(null, "password",
         JOptionPane.QUESTION_MESSAGE);

          if (user.equals("admin") && password.equals("1234")) {
              JOptionPane.showMessageDialog(null, "login ok");

          }else {
              JOptionPane.showMessageDialog(null, "login failed");
          }
    }
}
```