El contenedor *JDialog*

- *JDialog* es un contenedor que puede albergar componentes gráficos

- Su principal característica es que pueden ser "modales", es decir, con la capacidad de tomar el foco de la aplicación y que no se pueda realizar ninguna operación hasta que se cierren

- Los utilizaremos para diseñar ventanas "emergentes" que requieran operaciones de entrada y salida de datos (inserción, actualización, etc.)

El contenedor *JDialog*

- Al crear un nuevo *JDialog,* NetBeans genera código que incluye un método main() y un constructor de este tipo:

```
public NewJDialog(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }
```

- Para el proyecto de prácticas vamos a crear los *JDialog* con un constructor sin parámetros, así que eliminamos el código que no necesitamos:

```
public NewJDialog( ) {
    initComponents();
    }
```

Y también eliminamos el método main()

El contenedor *JDialog*

- Desde un controlador, para abrir un *JDialog* con la propiedad "modal", lo haremos así:

```
// vCRUDMonitor es un objeto de una clase de tipo JDialog
vCRUDMonitor = new VistaCRUDMonitor();
vCRUDMonitor.setLocationRelativeTo(c: null);
vCRUDMonitor.setModalityType(type: Dialog.ModalityType.APPLICATION_MODAL);
vCRUDMonitor.setResizable(resizable: false);
vCRUDMonitor.setVisible(b: true);
```

JOptionPane con *JDialog*

- Para que una ventana *JOptionPane* se sitúe delante de una ventana modal para mostrar, por ejemplo, un mensaje de información o de error, es necesario pasar el objeto de la ventana modal como primer parámetro del método *JOptionPane*
- Un ejemplo de método para mostrar mensajes en nuestra clase "vistaMensajes" podría ser:

```
public void Mensaje(Component C, String tipoMensaje, String texto) {
  switch(tipoMensaje) {
    case "info":
       JOptionPane.showMessageDialog(C, texto, "Información",
                                              JOptionPane.INFORMATION_MESSAGE);
       break;
     }
   }
```

- Llamada desde un controlador:

```
vMensaje = new VistaMensajes();

Si queremos que el JOptionPane se ponga por encima de la vista "vCRUDMonitor"
vMensaje.Mensaje(vCRUDMonitor, "info", "Mensaje que se quiere mostrar");

Si no es necesario indicar la vista "padre", solo es necesario enviar "null" como primer parámetro
vMensaje.Mensaje(null, "info", "Mensaje que se quiere mostrar");
```

JOptionPane con *JDialog*

- Para mostrar una ventana de diálogo con diferentes opciones, la clase *JOptionPane* dispone del método *showConfirmDialog*

- Por ejemplo, se puede mostrar una ventana con tres opciones: "YES", "NO" y "CANCEL"

```
public int Dialogo(Component C, String texto) {
       int opcion = JOptionPane.showConfirmDialog(C, texto,
                        "Atención", JOptionPane.YES_NO_CANCEL_OPTION,
                        JOptionPane.WARNING_MESSAGE);
       return opcion;
   }
```

- Y capturar los eventos del ratón con:
  - **JOptionPane.YES_OPTION**, si se pulsa el botón "YES"
  - **JOptionPane.NO_OPTION**, si se pulsa el botón "NO"
  - **JOptionPane.CANCEL_OPTION**, si se pulsa el botón "CANCEL"
  - **JOptionPane.CLOSED_OPTION**, si se cierra la ventana sin seleccionar ninguna opción

Look and Feel (L&F)

- El *look and feel* (aspecto y comportamiento) es la forma en que los componentes de Swing (*JLabel*, *JButton*, *JTextField*, *JTable*, *JComboBox*, etc.) se muestran dentro de una interfaz de usuario.

- *L&F* puede modificar el fondo, el tipo de letra, bordes, colores y, en general, el aspecto de los componentes

- Por defecto, el tema que utiliza el IDE de *Netbeans* en sus aplicaciones es *Nimbus*

- Para que la aplicación tenga el aspecto *Nimbus*, simplemente añadiremos este código inmediatamente después del *main()*

```
try {
   UIManager.setLookAndFeel( new NimbusLookAndFeel() );
} catch(Exception ex) {
   System.err.println( "Mensaje de error);
}
```

Look and Feel (L&F)

- *FlatLaf* es un moderno Look and Feel multiplataforma de código abierto para aplicaciones de escritorio que utilizan *Swing*
- Presenta un aspecto limpio, sencillo y elegante (sin sombras ni degradados)
- Tiene temas como *Light*, *Dark*, *IntelliJ* y *Darcula*

- Para utilizarlos, simplemente debemos poner su dependencia en el fichero pom.xml

```
<dependency>
   <groupId>com.formdev</groupId>
   <artifactId>flatlaf</artifactId>
   <version>3.2.5</version>
</dependency>
```

- Y asignarle a a *UIManager* nuestro tema preferido para la aplicación

```
try {
   UIManager.setLookAndFeel( new FlatDarculaLaf() );
} catch(Exception ex) {
   System.err.println( "Mensaje de error);
}
```

Pulsando aquí CTRL+BarraEspaciadora aparecerán todos los temas disponibles