PRACTICA INTERNAL FRAME

Duración estimada: 200 min.

Objetivos:

- Trabajar con menús en aplicaciones
- Trabajar con ventanas internas en aplicaciones
- Diseñar aplicaciones con menú
- Trabajar con cuadros de diálogo

Recursos necesarios:

- Apuntes dados en clase.
- Guión de la práctica
- Internet.

Introducción:

1. **Menús:**

Java Swing nos provee ciertos componentes para crear una barra de Menú, en ella podemos combinar diferentes elementos con un mismo fin, proveer las opciones necesarias para trabajar con el sistema.

- **JmenuBar:** Es el elemento principal cuando vamos a crear menús, ya que provee la barra donde se alojaran cada uno de los items u opciones deseadas.

```
barraMenu = new JMenuBar();
barraMenu.add(menuArchivo);
 setJMenuBar(barraMenu);
```

- **Jmenu:** Si bien el elemento anterior permite crear una Barra donde alojar el resto de componentes, el Jmenu es quien contiene dichos componentes, a este se le agregan el resto de opciones, podemos asociarlo con un contenedor el cual aloja otros elementos como botones, etiquetas, campos entre otros..... el JMenu permite agregar los elementos o items correspondientes, así como otros Jmenus.....

```
menuArchivo = new JMenu();
menuArchivo.add(menuItemNuevo);
menuArchivo.addSeparator();
```

- **JmenuItem:** Representan Items u opciones del menú, cuando creamos un JMenu decimos que tipo de opciones puede contener y al crear un JMenuItem decimos cuales son las opciones para ese menú en especifico, por ejemplo el Menú "Archivo", contendrá los items "Abrir", "Guardar", "Nuevo", "Principal", etc....

```
menuItemNuevo = new JMenuItem();
menuItemNuevo.setText("Nuevo");
menuArchivo.add(menuItemNuevo);
```

- **JcheckBoxMenuItem :** Este compontes es otro item que almacena el JMenu, permite vincular casillas de verificación o Checkbox, muy útil cuando vamos a parametrizar mas de una opción o característica de nuestro sistema...

```
jCheckMenu1 = new JCheckBoxMenuItem("Check1");
 menuOpciones.add(jCheckMenu1);
```

- **JradioButtonMenuItem:** Este componente es similar al anterior, la diferencia es que permite vincular componentes RadioButton los cuales brindan opciones de selección única, por ejemplo si en un editor de texto queremos un único tipo de letra, podemos usar este componente....

```
RadioButtonMenu1 = new JRadioButtonMenuItem("Radio1");
grupoRadios.add(jRadioButtonMenu1);
menuOpciones.add(jRadioButtonMenu1);
```

- **JpopupMenu:**Por ultimo tenemos el JPopupMenu, a diferencia de los anteriores, este componente no es contenido en la Barra de Menú, sino que se asocia al contenedor principal que para nuestro ejemplo es un JPanel (del cual hablaremos en el siguiente articulo), permite brindar opciones emergentes o popup con tan solo dar click derecho sobre algún área del panel..... el JPopup funciona también como un contenedor similar al Jmenu....

```
menuEmergente = new JPopupMenu();
itemEmergente1.setText("Opcion Emergente 1");
menuEmergente.add(itemEmergente1)
```

2. Frames internos:

El **JinternalFrame** es una ventana especial que ni es ventana ni es nada. De hecho, no hereda de Window. En realidad es un componente java que se dibuja dentro de otro componente, pero adornado con la barra de título de una ventana y sus botones de maximizar, minimizar y cerrar. Puede incluso arrastrarse y modificarse de tamaño siempre que se meta dentro del componente adecuado. El sitio bueno para meter los JinternalFrame es el **JDesktopPane**. Este panel admite JinternalFrame y sabe manejarlos.

Con este código el JinternalFrame se comportará como una ventana, pero que no puede salirse del JDesktop que la contiene.

Con la clase JinternalFrame se puede mostrar un Jframe como una ventana dentro de otra ventana. Para crear un frame interno que parezca un diálogo sencillo, se pueden utilizar los cuadros de diálogo..

Los frames internos utilizan la decoración de ventana del aspecto y comportamiento Metal. Sin embargo, la ventana que los contiene tiene decoración de aspecto y comportamiento nativo (en este caso, Motif).

El código para utilizar frames internos es similar en muchas formas al código para utilizar frames normales Swing. Los frames internos no son ventanas, por lo que de alguna forma son diferentes de los frames. Por ejemplo, debemos añadir un frame interno a un contenedor (normalmente un JDesktopPane). Un frame interno no genera eventos window; en su lugar, las acciones del usuario que podrían causar que un frame dispara eventos windows hacen que en un frame interno se disparen eventos "internal frame".

Como los frames internos se han implementado con código independiente de la plataforma, ofrecen algunas características que los frames no pueden ofrecer:

- Una de esas características es que los frames internos ofrecen más control sobre su estado y capacidades. Se puede minimizar o maximizar un frame interno, a través del código.
- También se puede especificar el icono que va en la barra de título del frame interno.

- Incluso podemos especificar si el frame tiene soporte de decoración de ventanas, redimensionado, minimización, cerrado y maximización.
- Otra característica es que los frames internos se han diseñado para trabajar con paneles por capas. El API JinternalFrame contiene métodos como ***moveToFront*** que funcionan sólo si el contenedor de un frame interno es un layeredpane.

- **Reglas de utilización de Frames Internos**
  - Se debe seleccionar el tamaño del frame interno. Si no se selecciona el tamaño del frame interno, tendrá tamaño cero y nunca será visible. Se puede seleccionar el tamaño utilizando uno de estos métodos:***setSize,pack o setBounds***.
  - Como regla, se debe seleccionar la posición del frame interno. Si no se selecciona la localización, empezará en 0,0 (la esquina superior izquierda de su contenedor). Se pueden utilizar los métodos ***setLocation o setBounds*** para especificar la esquina superior izquierda del frame interno en relación a su contenedor.
  - Para añadir componentes a un frame interno, se añaden al panel de contenidos del propio frame interno.
  - Un frame interno se debe añadir a un contenedor. Si no lo añadimos a un contenedor (normalmente un JDesktopPane), el frame interno no aparecerá.
  - Normalmente no se tiene que llamar a show o setVisible para los frames internos.
  - Los frames internos disparan eventos "internal frame", no eventos "window". El manejo de eventos "internal frame" es casi idéntico al manejo de eventos "window".

- **Crear un Frame Interno**

Constructor	Propósito
JInternalFrame() **JInternalFrame(String)** **JInternalFrame(String, boolean)** **JInternalFrame(String, boolean, boolean)** **JInternalFrame(String, boolean, boolean, boolean)** **JInternalFrame(String, boolean, boolean, boolean, boolean)**	Crea un ejemplar de **JInternalFrame**. El primer argumento especificar el título (si existe) a mostrar por el frame interno. El resto de los argumentos especifican si el frame interno debería contener decoración permitiendo al usuario que redimensione, cierre, maximice y minimice el frame interno (por este orden). El valor por defecto para cada argumento booleano es **false**, lo que significa que la operación no está permitida.
Métodos de la clase**JOptionPane**: • **showInternalConfirmDialog** • **showInternalInputDialog** • **showInternalMessageDialog** • **showInternalOptionDialog**	Crea un **JinternalFrame** que simila un diálogo.

- **Añadir Componentes a un Frame Interno**

Método	Propósito
void setContentPane(Container) **Container getContentPane()**	Selecciona u obtiene el panel de contenido del frame interno, que generalmente contiene todo e GUI del frame interno, con la excepción de la barra de menú y las decoraciones de la ventana.
void setMenuBar(JMenuBar) **JMenuBar getMenuBar()**	Selecciona u obtiene la barra de menú del frame interno. Observa que estos nombres de método no contienen "J", al contrario que sus métodos equivalentes de **JFrame**. En las siguientes versiones de Swing y del JDK 1.2, **JinternalFrame** añadira **setJMenuBar** y **getJMenuBar**, que se deberían utilizar en vez de los métodos existentes.

- **Especificar el Tamaño y la Posición del Frame Interno**

Método	Propósito
void pack()	Dimensiona el frame interno para que sus componentes tenga sus tamaños preferidos.
void setLocation(Point) **void setLocation(int, int)**	Seleciona la posición del frame interno. (Heredada de **Component**).
void setBounds(Rectangle) **void setBounds(int, int, int, int)**	Explicitámente selecciona el tamaño y la localización del frame interno (Heredada de **Component**).
void setSize(Dimension) **void setSize(int, int)**	Explicitámente selecciona el tamaño del frame interno. (Heredada de **Component**).

- **Realizar Operaciones de Ventana sobre el Frame Interno**

Método	Propósito
void setDefaultCloseOperation(int) **int getDefaultCloseOperation()**	Selecciona u obtiene lo que hace el frame interno cuando el usuario intenta "cerrar" el frame. El valor por defecto es **HIDE_ON_CLOSE**. Otros posibles valores son **DO_NOTHING_ON_CLOSE** y **DISPOSE_ON_CLOSE**.
void addInternalFrameListener(InternalFrameListener) **void removeInternalFrameListener(InternalFrameListener)**	Añade o elimina un oyente de "internal frame" (**JinternalFrame** es equivalente a un oyente de "window").
void moveToFront() **void moveToBack()**	Si el padre del frame interno es un layeredpane, mueve el frame interno adelante o detrás (respectivamente) por sus capas).
void setClosed(boolean) **boolean isClosed()**	Selecciona u obtiene si el frame interno está cerrado actualmente.
void setIcon(boolean) **boolean isIcon()**	Minimiza o maximiza el frame interno, o determina si está minimizado actualmente.
void setMaximum(boolean) **boolean isMaximum()**	Maximiza o restaura el frame interno o determina si está maximizado.
void setSelected(boolean) **boolean isSelected()**	Selecciona u obtiene si el frame interno esta actualmente "seleccionado" (activado).

- **Controlar la decoración y las capacidades de la ventana**

Método	Propósito
void setFrameIcon(Icon) **Icon getFrameIcon()**	Seleciona u obtiene el icono mostrado en la barra de título del frame interno (normalmente en la esquina superior izquierda).
void setClosable(boolean) **boolean isClosable()**	Selecciona u obtiene si el usuario puede cerrar el frame interno.
void setIconifiable(boolean) **boolean isIconifiable()**	Selecciona u obtiene si el frame interno puede ser minimizado.
void setMaximizable(boolean) **boolean isMaximizable()**	Selecciona u obtiene si el usuario puede maximizar el frame interno.
void setResizable(boolean) **boolean isResizable()**	Selecciona u obtiene si el frame interno puede ser redimensionado.
void setText(String) **String getText()**	Selecciona u obtiene el título de la ventana.
void setWarningString(String) **String getWarningString()**	Selecciona u obtiene cualquier string de aviso mostrado en la ventana.

Estudia los siguientes ejemplos

1. **Ejemplo1**: Estudia el siguiente codigo que se corresponde a una aplicación que crea ventanas internas.

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JDesktopPane;
import javax.swing.JFrame;
import javax.swing.JInternalFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

/**
 * Internal Frames Demo
 *
 * @version $Id: JIFrameDemo.java,v 1.4 2003/07/15 01:46:47 ian Exp $
 */
```

```
public class InternalVentana {
 /* Main View */
 public static void main(String[] a) {
  final JFrame jf = new JFrame("Demostracion Ventana Interna");
  Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
  screenSize.width -= 42;
  screenSize.height -= 42;
  jf.setSize(screenSize);
  jf.setLocation(20, 20);

  JMenuBar mb = new JMenuBar();
  jf.setJMenuBar(mb);
  JMenu fm = new JMenu("Archivo");
  mb.add(fm);
  JMenuItem mi;
  fm.add(mi = new JMenuItem("Salir"));

  mi.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    System.exit(0);
   }
  });

  JDesktopPane dtp = new JDesktopPane();
  jf.setContentPane(dtp);

  JInternalFrame mboxFrame = new JInternalFrame("Correo", true,
    true, true, true);
  JLabel reader = new JLabel("LeerCorreo");
  mboxFrame.setContentPane(reader);
  mboxFrame.setSize(400, 300);
  mboxFrame.setLocation(50, 50);
  mboxFrame.setVisible(true);
  dtp.add(mboxFrame);

  JInternalFrame compFrame = new JInternalFrame("Crear Correo", true,
    true, true, true);
  JLabel composer = new JLabel("Debe crear correo aqui");
  compFrame.setContentPane(composer);
  compFrame.setSize(300, 200);
  compFrame.setLocation(200, 200);
  compFrame.setVisible(true);
  dtp.add(compFrame);
```

```
  JInternalFrame listFrame = new JInternalFrame("Usuarios", true, true, true,
      true);
  JLabel list = new JLabel("Lista de usuarios");
  listFrame.setContentPane(list);
  listFrame.setLocation(400, 400);
  listFrame.setSize(500, 200);
  listFrame.setVisible(true);
  dtp.add(listFrame);

  jf.setVisible(true);
  jf.addWindowListener(new WindowAdapter() {
   public void windowClosing(WindowEvent e) {
    jf.setVisible(false);
    jf.dispose();
    System.exit(0);
   }
  });
 }
}
```

2. **Ejemplo2: Ejecuta la siguiente aplicación que muestra una ventana principal con su cuadro de diálogo**

```
public class VentanaPrincipal extends JFrame implements ActionListener {
      private Container contenedor;/**declaramos el contenedor*/
      JButton botonCambiar;/**declaramos el objeto Boton*/
      JLabel labelTitulo;/**declaramos el objeto Label*/
      private VentanaPrincipal miVentanaPrincipal;
      public VentanaPrincipal(){
      /**permite iniciar las propiedades de los componentes*/
      iniciarComponentes();
      /**Asigna un titulo a la barra de titulo*/
      setTitle("CoDejaVu : JFrame VentanaPrincipal");
      /**tamaño de la ventana*/
      setSize(300,180);
      /**pone la ventana en el Centro de la pantalla*/
      setLocationRelativeTo(null);
}

/**
* @param miVentana
* Enviamos una instancia de la ventana principal
*/
public void setVentanaPrincipal(VentanaPrincipal miVentana) {
      miVentanaPrincipal=miVentana;
}

private void iniciarComponentes() {
      contenedor=getContentPane();/**instanciamos el contenedor*/
      /**con esto definmos nosotros mismos los tamaños y posicion
      * de los componentes*/
      contenedor.setLayout(null);
```

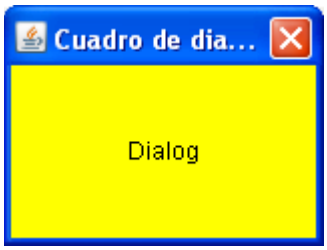
```
        /**Propiedades del boton, lo instanciamos, posicionamos y
        * activamos los eventos*/
        botonCambiar= new JButton();
        botonCambiar.setText("Iniciar");
        botonCambiar.setBounds(100, 80, 80, 23);
        botonCambiar.addActionListener(this);

 /**Propiedades del Label, lo instanciamos, posicionamos y
 * activamos los eventos*/
        labelTitulo= new JLabel();
        labelTitulo.setText("VENTANA PRINCIPAL");
        labelTitulo.setBounds(80, 20, 180, 23);

 /**Agregamos los componentes al Contenedor*/
        contenedor.add(labelTitulo);
        contenedor.add(botonCambiar);
 }
 /**Agregamos el evento al momento de llamar la otra ventana*/
 @Override
 public void actionPerformed(ActionEvent evento) {
        if (evento.getSource()==botonCambiar)
              {
        /*enviamos la instancia de la ventana principal para que
        * esta sea Padre de la ventana de dialogo*/
              VentanaConfirmacion miVentanaConfirmacion= new VentanaConfirmacion
        (miVentanaPrincipal,true);
        miVentanaConfirmacion.setVisible(true);
        }
        }
}
```

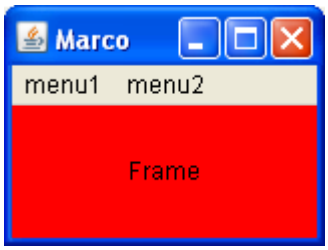
Jdialog:

```
public class VentanaConfirmacion extends JDialog{
private Container contenedor;
JLabel labelTitulo;

public VentanaConfirmacion(VentanaPrincipal miVentanaPrincipal, boolean modal){
/**Al llamar al constructor super(), le enviamos el
* JFrame Padre y la propiedad booleana que determina
* que es hija*/
        super(miVentanaPrincipal, modal);
        contenedor=getContentPane();
        contenedor.setLayout(null);
        //Asigna un titulo a la barra de titulo
        setTitle("CoDejaVu : JDialog VentanaConfirmacion");

        labelTitulo= new JLabel();
        labelTitulo.setText("VENTANA DE CONFIRMACION");
        labelTitulo.setBounds(20, 20, 180, 23);

        contenedor.add(labelTitulo);
        //tamaño de la ventana
        setSize(350,150);
        //pone la ventana en el Centro de la pantalla
        setLocationRelativeTo(null);
        }
}
```

Secuencia y desarrollo:

3. **Ejemplo1:** Implementa las siguientes ventanas en modo texto. Estudia las diferencias de cada una de ellas.





Creación de una ventana

```
import java.applet.*;
import java.awt.*;

public class Ventana extends Window {
        Ventana(Frame fr,int x,int y,int dx, int dy) {
                super(fr);
                Label lbl=new Label("Window",Label.CENTER);
                lbl.setBackground(Color.cyan);
                add("Center",lbl);
                setSize(dx,dy);
                setLocation(x,y);
                setVisible(true);
        }
}
```

Creación de un marco

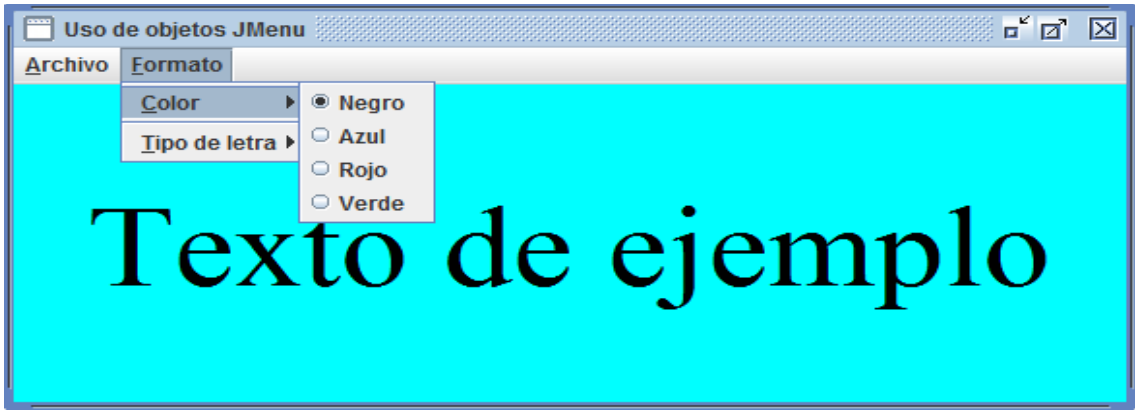
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Marco extends Frame implements WindowListener {
        Marco(String título,int x,int y,int dx,int dy) {
                super(título);
                setBackground(Color.red);
                Label lbl=new Label("Frame",Label.CENTER);
                setBackground(Color.red);
                add("Center",lbl);

                MenuBar menubar=new MenuBar();
                setMenuBar(menubar);
                Menu menu1=new Menu("menu1");
                menu1.add(new MenuItem("item 1.1"));
                menu1.add(new MenuItem("item 1.2"));
                menu1.add(new MenuItem("item 1.3"));
                Menu menu2=new Menu("menu2");
                menu2.add(new MenuItem("item 2.1"));
                menu2.add(new MenuItem("item 2.2"));
                menubar.add(menu1);
                menubar.add(menu2);

                setSize(dx,dy);
                setLocation(x,y);
                setVisible(true);
                addWindowListener(this);
        }
```

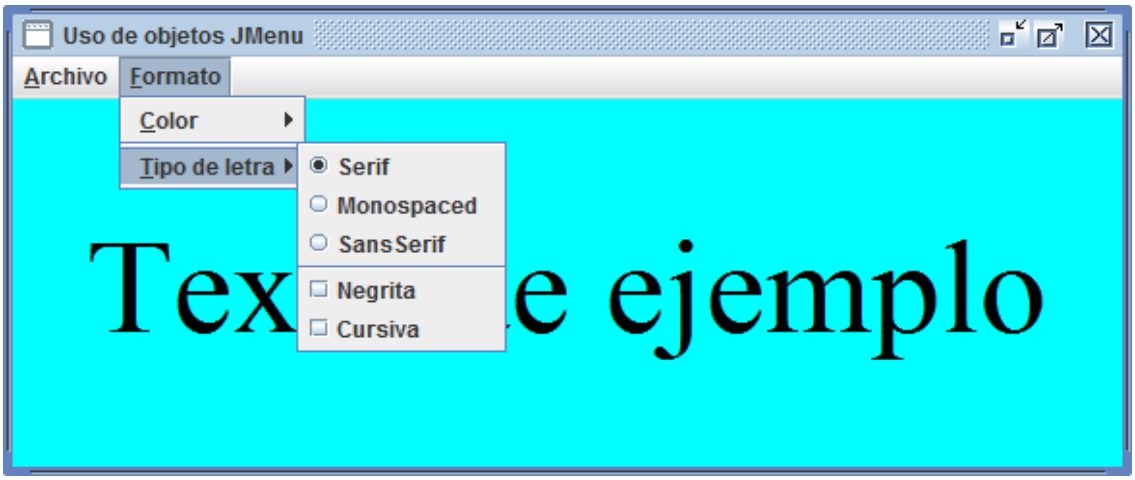
```
        /*------------ WindowListener ----------*/
    public void windowOpened(WindowEvent e){}
    public void windowActivated(WindowEvent e){}
    public void windowDeactivated(WindowEvent e){}
    public void windowIconified(WindowEvent e){}
    public void windowDeiconified(WindowEvent e){}
    public void windowClosed(WindowEvent e){}
        public void windowClosing(WindowEvent e){
            hide();
        }

}
```

Creación de un cuadro de diálogo

```
import java.applet.*;
import java.awt.*;

public class Dialogo extends Dialog {

        Dialogo(Frame madre,String título,boolean modal,int x,int y,int dx,int dy) {
            super(madre,título,modal);
            Label lbl=new Label("Dialog",Label.CENTER);
            lbl.setBackground(Color.yellow);
            add("Center",lbl);
            setSize(dx,dy);
            setLocation(x,y);
            setVisible();
        }
}
```

1. **Ejercicio1**: Implementa el siguiente ejercicio, de tal forma que, pulsando cada una de las opciones, tiene que cambiar el color y el tipo de letra asociado al texto que aparece en la ventana. Dentro del menú Archivo, añadir la opción de cambiar el color de fondo del documento.





- Añadir al menú **Archivo** la opción **Nuevo** para abrir una nueva ventana interna.