

- <https://www.dropbox.com/s/mbzz5m2aixbwcoq/SWINGAlumno.pdf?dl=0>
- Imágenes
- <https://www.dropbox.com/s/nbpf64idfc68l9/IMAGENES.zip?dl=0>
- GUI
- <https://www.dropbox.com/s/d74lidenatxqryh/Crear%20un%20GUI.pdf?dl=0>

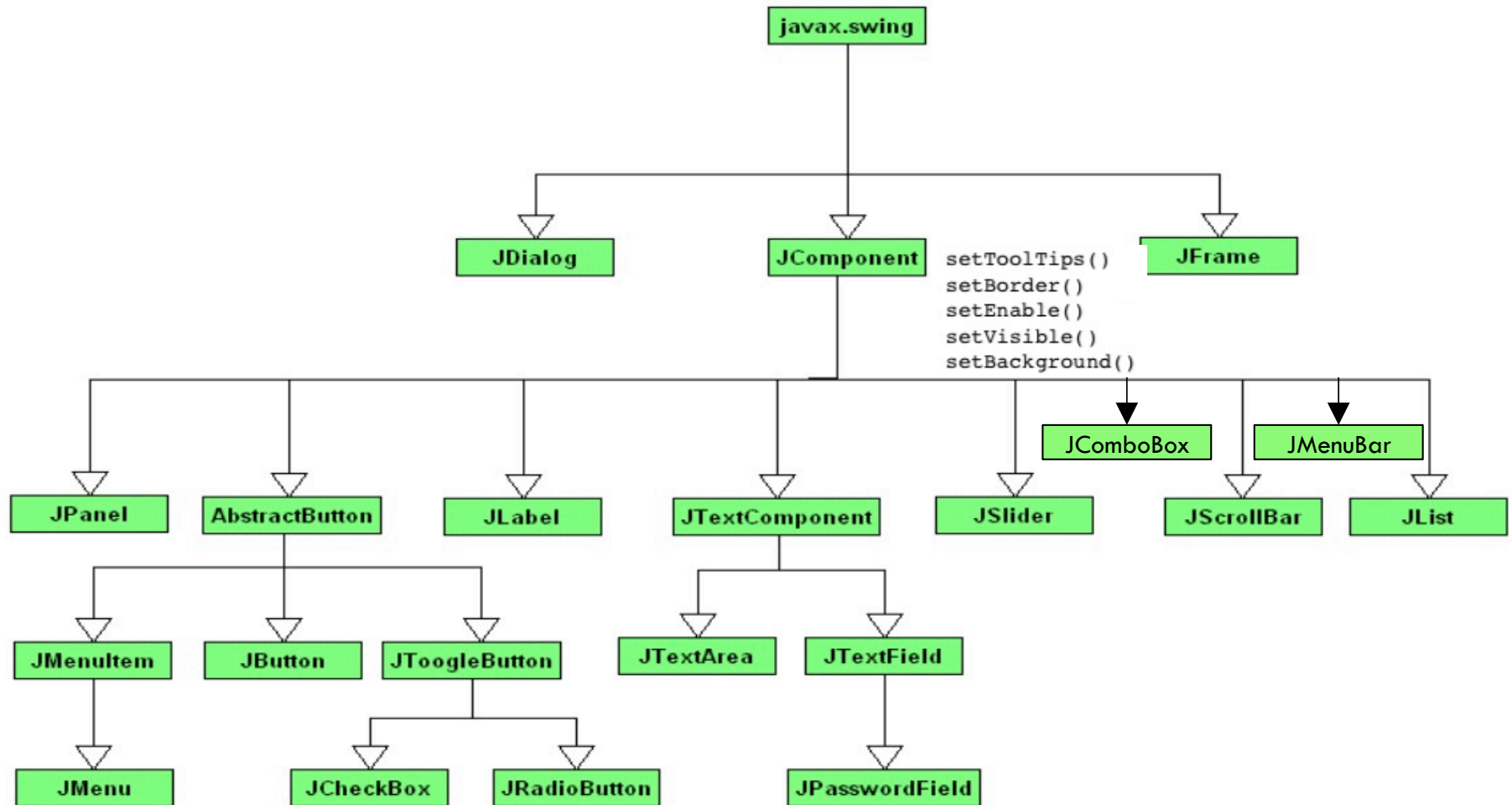
SWING

- AWT se apoya en los componentes proporcionados por el Sistema Operativo Destino, llamados comúnmente componentes pesados, que
 - ▣ Consume muchos recursos del sistema operativo
 - ▣ No funcionan de manera exactamente igual en cada máquina, lo que da problemas de portabilidad
 - ▣ Su Aspecto "*Look&Feel*" esta ligado al Sistema donde se ejecuta (se ve diferente en cada plataforma), y no puede cambiarse
 - ▣ Swing está escrito completamente en Java y no depende tanto del Sistema (los contenedores principales en Swing son pesados, el resto no):
 - Consume menos recursos del sistema operativo
 - Mejor y mayor portabilidad
 - El *Look & Feel* puede hacerse independiente de la plataforma (para que se vea igual en todas las plataformas) o no, y se puede seleccionar
- Swing ofrece una mucho mayor gama de componentes que AWT
- Swing amplía los métodos y eventos disponibles en AWT
- Swing amplía funcionalidades que AWT y ofrece otras nuevas

SWING

- La metodología de trabajo con *Swing* es prácticamente idéntica a la que estudiamos con AWT. Los *listeners* trabajan de la misma manera y el funcionamiento de los *layouts* sobre los *containers* también es el mismo.
- Todos los componentes heredan de *javax.swing.Jcomponent*
- *JFrame* será la base para la aplicación principal. *JDialog* construirá los diálogos (ventanas).
- El resto de clases serán componentes simples.
- Usar en todas las clases *import javax.swing.*;* y *import java.awt.*;*
- Los componentes que provee *Swing* son fácilmente identificables porque comienzan con el prefijo “J”. Por ejemplo: *JButton*, *TextField*, *JPanel*, *JFrame*, *JLabel*. Los *layouts* y los *listeners* son los mismos para AWT y para *Swing*.

Clases del paquete SWING



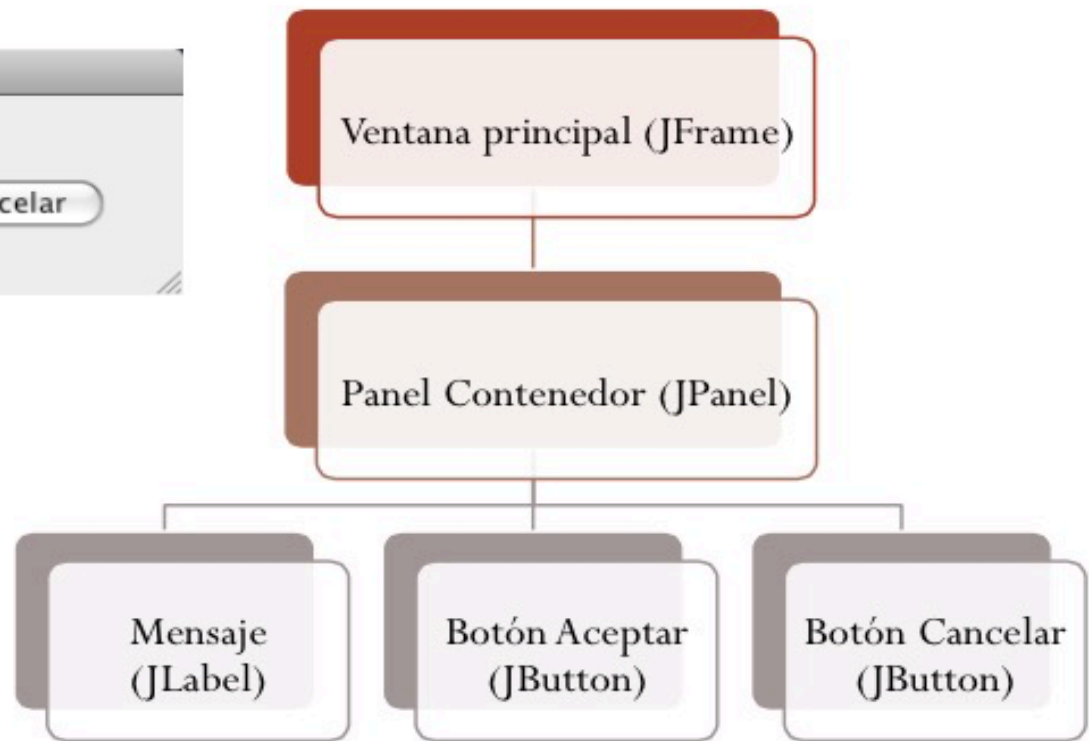
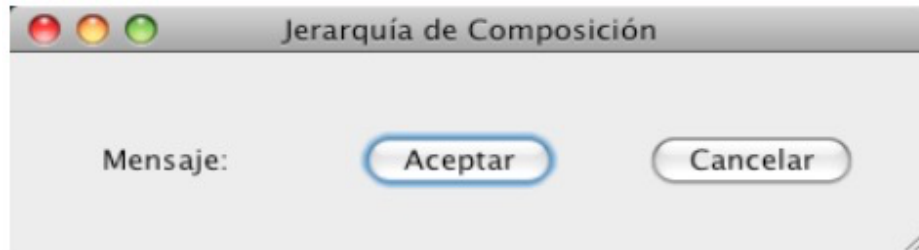
SWING:

Pasos básicos en la Construcción de un GUI

- ❑ Crear una nueva Clase para nuestra Ventana (extends JFrame) o instanciar un JFrame
- ❑ Crear los componentes de nuestra Interfaz
- ❑ Crear uno o varios contenedores Intermedios
- ❑ Asociar los componentes al Contenedor
- ❑ Hacer Visible la Ventana

SWING:

Ejemplo Jerarquía de una composición



SWING Menús

- ❑ Los menús han de ir en la ventana principal de la aplicación.
- ❑ Pueden ser de tres tipos:
 - ▣ Drop-Down son los que saldrán al hacer click en Archivo
 - ▣ Submenu: son aquellos que salen como un grupo de un elemento de menú
 - ▣ Contextuales: (clase JPopupMenu) son aplicables a la región en la que está localizado el puntero del ratón.
- ❑ Son las clases *JMenuBar*, *JMenu* y *JMenuItem*.

SWING Contenedores

- Clase *JToolBar* : Esta clase implemente una barra de herramientas, formada normalmente por botones o controles que incluyen iconos y que aparecen organizados como una fila o una columna dependiendo de la zona de la pantalla donde se coloque. Son botones de comando o conmutación. Se suelen emplear gráficos.
 - ▣ Métodos *setFlotable(boolean)* y *addSeparator()*
- Clase *JPanel* : Es un contenedor que agrupa componentes dentro de una ventana.
 - ▣ Los layouts permiten un correcto posicionamiento de los componentes.
- Clase *JTabbedPane*: Es un contenedor que permite tener varios componentes separador por pestañas.

- JPanel: Para añadir un JPanel lo primero que se debe de hacer es obtener uno de los objetos que forman el frame o ventana, **el contenedor de paneles**, “content pane”. Para ello se invocará el método `getContentPane` de la clase `JFrame`. El objeto que obtenemos es de tipo `Container`.
 - ▣ `Container[nombre_del_contentpane] = frame.getContentPane() ;`
 - ▣ `Panel pNorth = new Panel(new FlowLayout(FlowLayout.LEFT)); //AWT
add(pNorth, BorderLayout.NORTH);`
 - ▣ A continuación se invoca el método `add` del `Container` obtenido para añadir el panel, pasándole el propio panel al método:
 - ▣ `[nombre_del_contentpane].add(nombre_del_panel);`

```
//Obtengo el objeto contenedor del frame
Container contentpane = getContentPane();
//Se crea un objeto de tipo JPanel
JPanel panel = new JPanel();
//Se añade el panel al objeto contenedor del frame
contentpane.add(panel);
//Para que se pueda apreciar el panel
//pongo su color de fondo verde
panel.setBackground(Color.green);
```

Ejemplo

```
JPanel pGL = new JPanel();
Jpanel pCL = new JPanel();
...
final Container micontenedor = getContentPane();
pGL.setLayout(new GridLayout(1, 2));
pCL.setLayout(new CardLayout(14, 14));
micontenedor.setLayout(new BorderLayout());
micontenedor.add(pCL, BorderLayout.SOUTH);
micontenedor.add(pGL, BorderLayout.NORTH);
```

```

import javax.swing.*;import java.awt.event.*;
import java.awt.Color;import java.awt.Container;
class frame extends JFrame {
    public frame()
    { setTitle("Curso de Java. Paneles");
      setSize(300, 200);
      addMouseListener(new manejadorMouse());
      addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent e)
          {System.out.println("...Cerrando Ventana");
            System.exit(0);} });
      Container contentpane = getContentPane();//Obtengo el objeto contenedor del frame
      JPanel panel = new JPanel();//Se crea un objeto de tipo JPanel
      contentpane.add(panel);//Se añade el panel al objeto contenedor del frame
      panel.setBackground(Color.green);}
    //Definimos las clases que van a escuchar eventos
    class manejadorMouse extends MouseAdapter {
        public void mousePressed(MouseEvent e) {
            System.out.println("...Mouse presionado");}}
    public static void main(String[] args){
        JFrame frame=new frame();frame.setVisible(true);}}

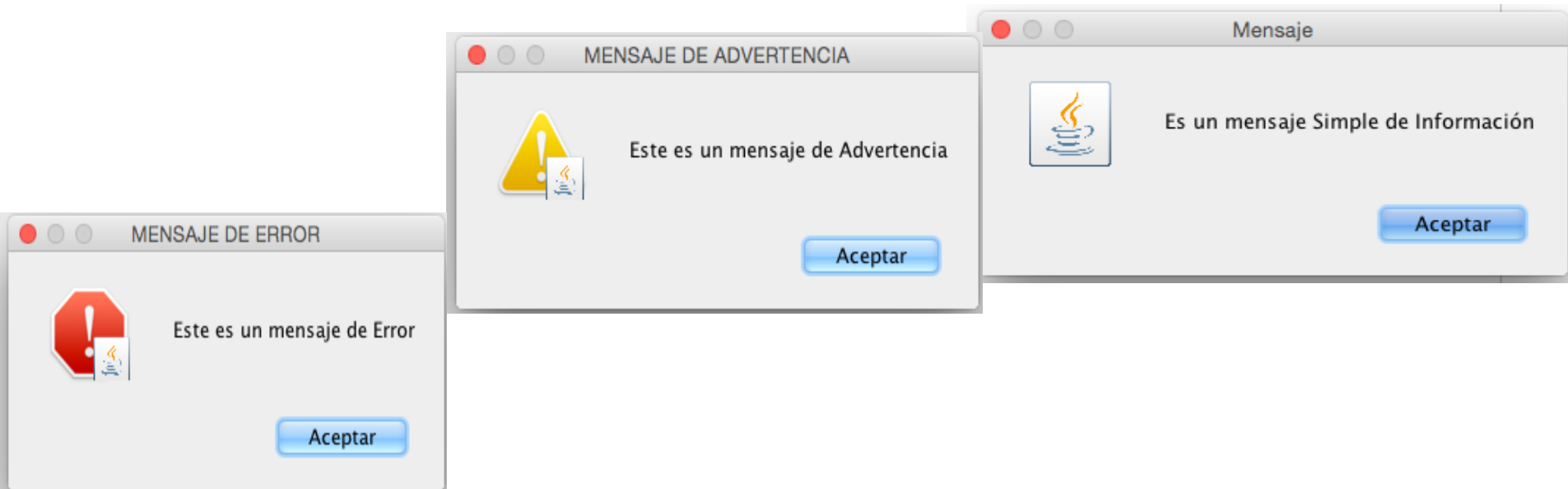
```

SWING JDialog

- Son ventanas mas limitadas que los Frames, y dependientes de estos, si se destruye el Frame, también lo hace el diálogo. Pueden ser:
 - ▣ No modales: No impiden interactuar con el Frame.
 - ▣ Modales: Impiden interactuar con el resto.
- Los más importantes son: JOptionPane y JFileChooser.
- JOptionPane
 - ▣ Permite adaptar y crear varias clases de diálogos, especificando por ejemplos los iconos, el título y texto de los diálogos.
 - **JOptionPane.showMessageDialog:** Genera una ventana de dialogo, con la cual podemos presentar al usuario un mensaje simple
 - **JOptionPane.showInputDialog:** Una ventana simple con un campo de texto para ingresar información
 - **JOptionPane.showConfirmDialog:** Una ventana de confirmación
 - **JOptionPane.showOptionDialog:** Con este podemos crear una ventana de dialogo con diferentes opciones .
- JFileChooser: Permite navegar por el sistema de ficheros, y seleccionar uno o varios ficheros.
Métodos importantes: `multiSelectionEnabled(boolean);` `getSelectedFile();`

SWING Diálogos

- JOptionPane: es una Clase que nos provee una serie de ventanas de dialogo predefinidas con el fin de facilitarnos algunos procesos de interacción con el usuario. Los iconos estándar son: *question*, *information*, *warning* y *error*.
 - ▣ **JOptionPane.showMessageDialog:** nos genera una ventana de dialogo, con la cual podemos presentar al usuario un mensaje simple, Ejemplo
`JOptionPane.showMessageDialog(null, "Este es un mensaje de Advertencia",
" MESAJE DE ADVERTENCIA", JOptionPane.WARNING_MESSAGE);`

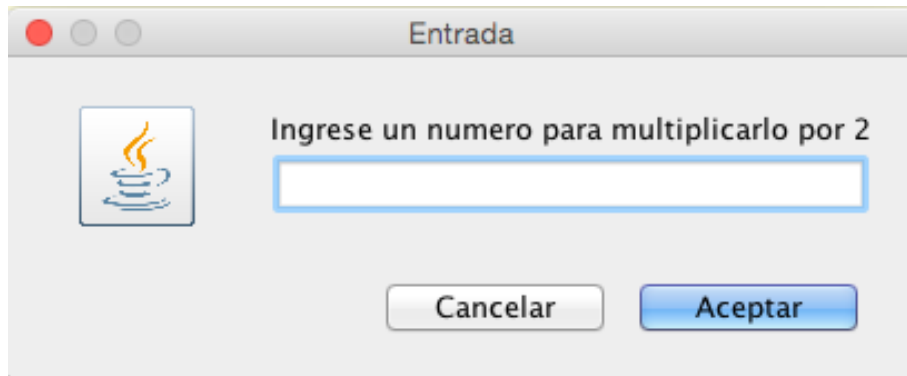


SWING Diálogos

- **JOptionPane.showInputDialog:** nos genera una ventana de dialogo, con la que presenta una ventana con un campo de texto para ingresar información, por defecto podemos obtener el dato ingresado mediante un String

■ Ejemplo:

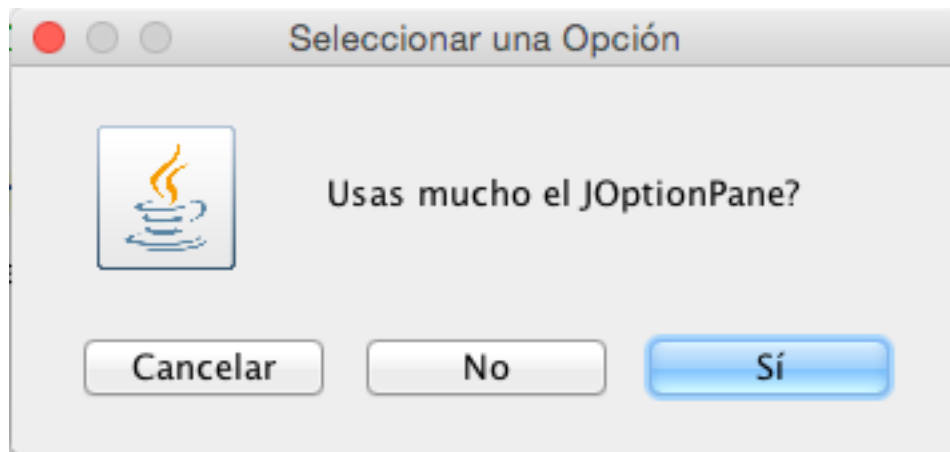
```
int numero=Integer.parseInt(JOptionPane.showInputDialog("Ingrese un" +  
    " numero para multiplicarlo por 2"));  
Object color = JOptionPane.showInputDialog(null,"Seleccione Un Color",  
    "COLORES", JOptionPane.QUESTION_MESSAGE, null,  
    new Object[] { "Seleccione", "Amarillo", "Azul", "Rojo" }, "Seleccione")  
String opcion= color.toString();
```



SWING Diálogos

- ▣ **JOptionPane.showConfirmDialog:** Este método nos brinda una ventana de confirmación donde por defecto se cargan 3 botones "Sí", "No", "Cancelar"

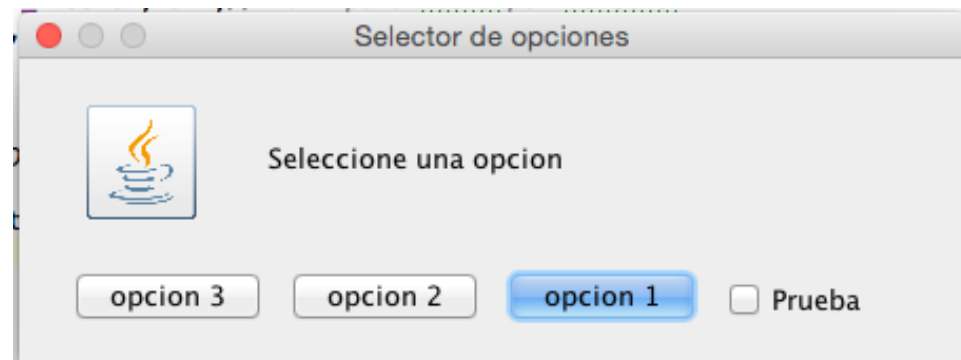
```
int resp=JOptionPane.showConfirmDialog(null,"Usas mucho el JOptionPane?");  
    if (JOptionPane.OK_OPTION == resp){  
        System.out.println("Selecciona opción Afirmativa");  
    } else{  
        System.out.println("No selecciona una opción afirmativa");  
    }
```



SWING Diálogos

- ▣ **JOptionPane.showOptionDialog:** Con este podemos crear una ventana de dialogo con diferentes opciones definidas en un array de objetos, podemos pasarle diferentes componentes gráficos:

```
JCheckBox chec=new JCheckBox("Prueba");  
int seleccion = JOptionPane.showOptionDialog( null,"Seleccione una opcion",  
    "Selector de opciones",JOptionPane.YES_NO_CANCEL_OPTION,  
    JOptionPane.QUESTION_MESSAGE,null,// null para icono por defecto.  
    new Object[] { "opcion 1", "opcion 2", "opcion 3",chec },"opcion 1");  
if (seleccion != -1)  
{System.out.println("seleccionada opcion " + (seleccion + 1)); }  
if (chec.isSelected()){  
    System.out.println("Selecciona el Chec");  
}
```



SWING Layouts

- Indican la forma de organizar los componentes dentro de un contenedor, determinando el tamaño y la posición. Para su uso:
 - ▣ Crear el contenedor.
 - ▣ Establecer el layout.
 - ▣ Agregar los componentes al contenedor.
- Tipos de layouts:
 - ▣ FlowLayout, BorderLayout, GridLayout, BoxLayout, GridBagLayout
- Por defecto:
 - ▣ Jpanel : FlowLayout
 - ▣ JFrame, Jdialog : BorderLayout

SWING : Tipos de Layout

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridLayout
- GridBagLayout
- GroupLayout
- SpringLayout

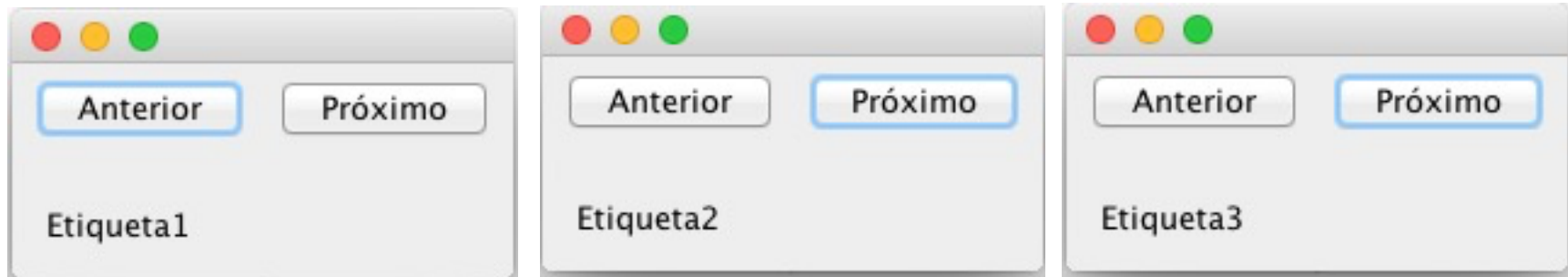


SWING Layouts

- **FlowLayout:** Es el más simple, los componentes añadidos a un contenedor se disponen en una o mas filas, de izquierda a derecha y de arriba a abajo.
- **BorderLayout:** Utiliza 5 áreas para colocar los componentes: Norte, Sur, Este, Oeste y Centro. Si alguna no se ocupa, se expande la contigua.
- **GridLayout:**
 - El controlador se crea con un determinado numero de filas y columnas.
 - Los componentes se sitúan de forma secuencial, de izquierda a derecha y de arriba a abajo.
 - El tamaño de las celdas es idéntico.
- **BoxLayout:**
 - Permite organizar los componentes en una línea horizontal o vertical, sin dejar espacio entre los componentes.
- **CardLayout:** Permite visualizar los componentes diferentes en plazos de tiempo diferentes, es decir, que visualiza sólo un componente a la vez.
- **GridBagLayout:** divide el contenedor **JPanel** como si fuera una rejilla de celdas, las celdas no tienen por qué tener el mismo tamaño. Cada componente puede ocupar una celda entera o sólo una parte. Si el componente va a ocupar un tamaño menor que la celda, puede estar centrado o alineado a algún borde de la celda. También es posible que un componente ocupe varias celdas.

SWING : Layout

- CardLayout permite visualizar los componentes diferentes en plazos de tiempo diferentes, es decir, que visualiza sólo un componente a la vez. Sin embargo el componente visualizado se puede cambiar.



SWING : Layout

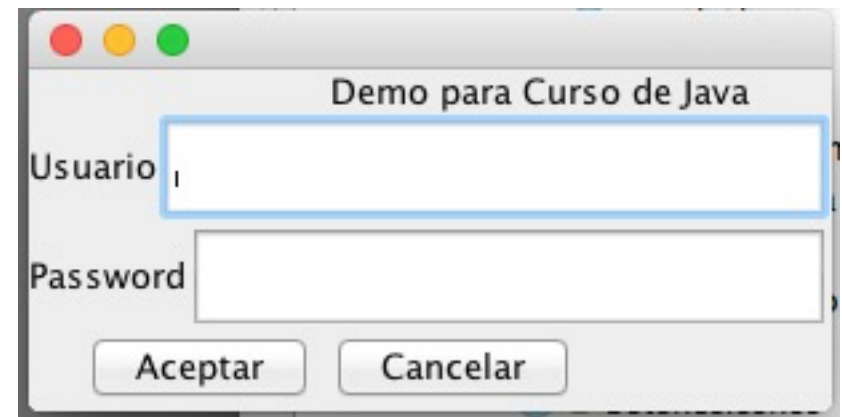
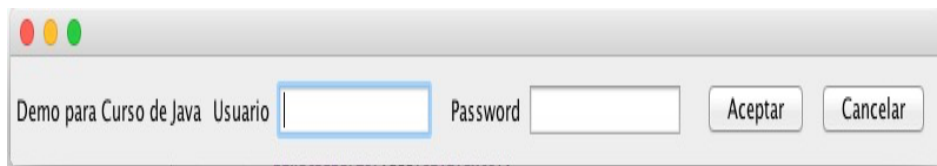
```
□ public class DemoCardLayout extends JFrame {  
    JLabel I1 = new JLabel("Etiqueta1");  
    JLabel I2 = new JLabel("Etiqueta2");  
    JLabel I3 = new JLabel("Etiqueta3");  
    JLabel I4 = new JLabel("Etiqueta4");  
    JLabel I5 = new JLabel("Etiqueta5");  
    JPanel NPB = new JPanel();  
    JPanel p1 = new JPanel();  
    Panel p = new Panel(new GridLayout(2, 1));  
    public DemoCardLayout() {  
        final Container micontenedor = getContentPane();  
        NPB.setLayout(new GridLayout(1, 2));  
        p1.setLayout(new CardLayout(14, 14));  
        micontenedor.setLayout(new BorderLayout());  
        micontenedor.add(p1, BorderLayout.SOUTH);  
        micontenedor.add(NPB, BorderLayout.NORTH);  
        JButton N = new JButton("Próximo");  
        JButton P = new JButton("Anterior");  
        NPB.add(P);  
        NPB.add(N);  
    }  
}
```

SWING : Layout

```
□ P.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        CardLayout cardLayout = (CardLayout) p1.getLayout();  
        cardLayout.previous(p1);  
    }  
});  
N.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        CardLayout cardLayout = (CardLayout) p1.getLayout();  
        cardLayout.next(p1);  
    }  
});  
  
p1.add( l1);  
p1.add( l2);  
p1.add( l3);  
p1.add( l4);  
p1.add( l5);  
micontenedor.setVisible(true);  
add(p);  
pack();  
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
setVisible(true);  
  
}
```

SWING : Layout

- BorderLayout, es un Layout parecido al FlowLayout, la diferencia es simple, FlowLayout ubica todos los componentes solo de forma horizontal, mientras que BorderLayout los ubica, tanto horizontal como vertical. Ejemplo
 - ▣ La idea general, es crear un JFrame general, el cual tendrá 3 paneles y una etiqueta, Estos estarán ordenados con BorderLayout en forma vertical.
 - ▣ Cada panel tendrán cajas de textos y etiquetas. Estos estarán ordenados con BorderLayout en forma horizontal.



```
□ public class DemoBoxLayout{
    private JPanel panelSuperior, panelMedio, panelInferior;
    private JLabel etiqueta1, etiqueta2, etiqueta3;
    private JTextField cajaTexto;
    private JPasswordField cajaPass;
    private JButton botonAceptar, botonCancelar;
    private JFrame frame;

    public void construyePanelSuperior(){
        panelSuperior = new JPanel();
        etiqueta2 = new JLabel("Usuario");
        cajaTexto = new JTextField(10);
        panelSuperior.setLayout(new BoxLayout(panelSuperior, BoxLayout.X_AXIS));
        panelSuperior.add(etiqueta2);
        panelSuperior.add(cajaTexto);
    }
}
```

```
□ public void construyePanelMedio(){
    panelMedio=new JPanel();
    etiqueta3= new JLabel("Password");
    cajaPass = new JPasswordField(10);
    panelMedio.setLayout(new BoxLayout(panelMedio, BoxLayout.X_AXIS));
    panelMedio.add(etiqueta3);
    panelMedio.add(cajaPass);
}

public void construyePanelInferior(){
    panelInferior=new JPanel();
    botonAceptar=new JButton("Aceptar");
    botonCancelar=new JButton("Cancelar");
    panelInferior.setLayout(new BoxLayout(panelInferior, BoxLayout.X_AXIS));
    panelInferior.add(botonAceptar);
    panelInferior.add(botonCancelar);
}
```




```
public void construyeVentana(){  
    frame = new JFrame();  
    etiqueta1 = new JLabel("Demo para Curso de Java");  
    frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS) );  
    frame.add(etiqueta1);  
    frame.add(panelSuperior);  
    frame.add(panelMedio);  
    frame.add(panelInferior);  
    frame.pack();  
    frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  
    frame.setVisible(true);  
}  
  
public DemoBoxLayout(){  
    construyePanelSuperior();  
    construyePanelMedio();  
    construyePanelInferior();  
    construyeVentana();  
}  
  
public static void main (String [] inforux){  
    new DemoBoxLayout();  
}  
}
```

SWING Clase JButton

- Es un botón que puede contener texto, gráficos, o ambos.

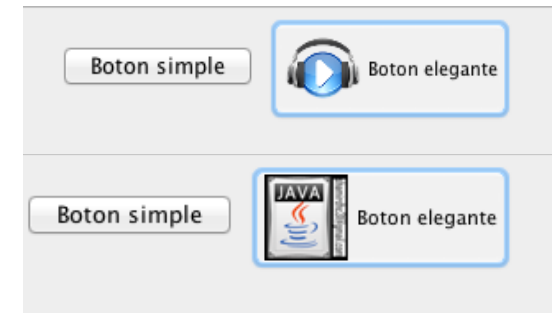
ACEPTAR

- Fijar el texto siempre centrado, en caso de contener una imagen, ha de ir a la izquierda o encima del texto.
- Métodos importantes:
 - ▣ `setText("Texto");`
`setToolTipText("Tooltip");`
`setBackground(Color.color);`
`setForeground(Color.color);`
`setIcon(new ImageIcon("ruta"));`
`setFont(new Font("tipo", estilo, tamaño));`
`setBounds(new Rectangle(posX,posY,tamX,tamY));` //redimensiona el componente
 - ▣ Y sus correspondientes `get`.



SWING Clase JButton

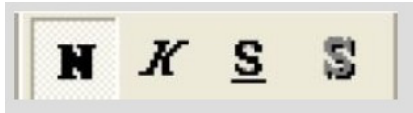

```
□ public PruebaJBoton()  
{ super( "Prueba de botones" );  
  Container c = getContentPane();  
  c.setLayout( new FlowLayout() )  
  botonSimple = new JButton( "Boton simple" );  
  c.add( botonSimple );  
  Icon icono1 = new ImageIcon( "icono1.gif" );  
  Icon icono2 = new ImageIcon( "icono2.gif" );  
  botonElegante = new JButton("Boton elegante",icono1);  
  botonElegante.setRolloverIcon( icono2 );  
  //agregamos el boton al panel de contenido  
  c.add( botonElegante );  
  ManejadorBoton manejador = new ManejadorBoton();  
  botonElegante.addActionListener( manejador );  
  botonSimple.addActionListener( manejador );  
  setSize( 300, 100 );  
  setVisible(true);  
} // fin del constructor de PruebaBoton
```



*se utiliza el metodo
setRolloverIcon heredado de
AbstractButton para
especificar la imagen que
aparece cuando el raton se
posiciona sobre el boton*

```
□ public static void main( String args[] )
{
    PruebaBoton ap = new PruebaBoton();
    ap.addWindowListener(
        new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {System.exit( 0 ); } // fin del método windowClosing
            } // fin de la clase interna anónima
    ); // fin de addWindowListener
} // fin de main
// clase interna para el manejo de eventos de botón
private class ManejadorBoton implements ActionListener {
    public void actionPerformed((ActionEvent e )
    {JOptionPane.showMessageDialog( null, "Usted oprimio:" + e.getActionCommand() );
    }}
}
```

SWING Clase JToggleButton

- **JToggleButton:** Es un botón que representa dos estados (On y Off). Mismas características que el JButton.
Puede emplearse como dos tipos de opciones.
- Independientes (Checkboxes). 
- Exclusivas (RadioButton). 
- Mismos métodos que JButton, pero añadiendo algunos nuevos.
 - `isSelected();`
 - `setSelected(boolean);`

SWING Class JCheckBox

□ Clase JCheckBox

- Es un control que representa dos estados (On y Off).
- Métodos `isSelected()` y `setSelected(boolean)`



□ Clase JRadioButton

- Permiten seleccionar **una** única opción dentro de un conjunto de opciones relacionadas.

```
JRadioButton pajaroButton = new JRadioButton("pajaro");
pajaroButton.setSelected(true);
JRadioButton gatoButton = new JRadioButton("gato");
JRadioButton perroButton = new JRadioButton("perro");
JRadioButton ConejoButton = new JRadioButton("conejo");
JRadioButton caballoButton = new JRadioButton("caballo");
//creamos un grupo y los añadimos
```

```
ButtonGroup group = new ButtonGroup();
group.add(pajaroButton);
group.add(gatoButton);
group.add(perroButton);
group.add(conejoButton);
group.add(caballoButton);
```

```
RadioListener myListener = new RadioListener();
```

```
pajaroButton.addActionListener(myListener);
```

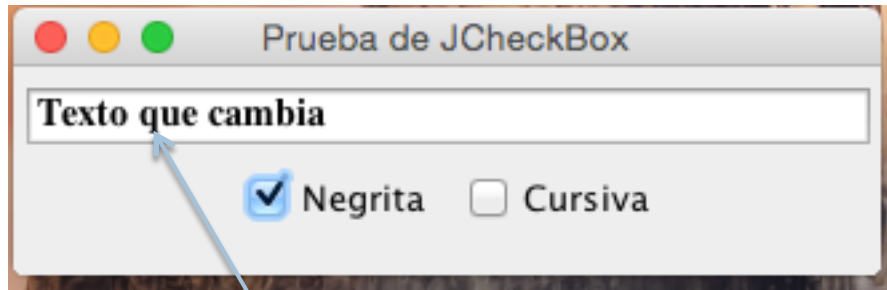
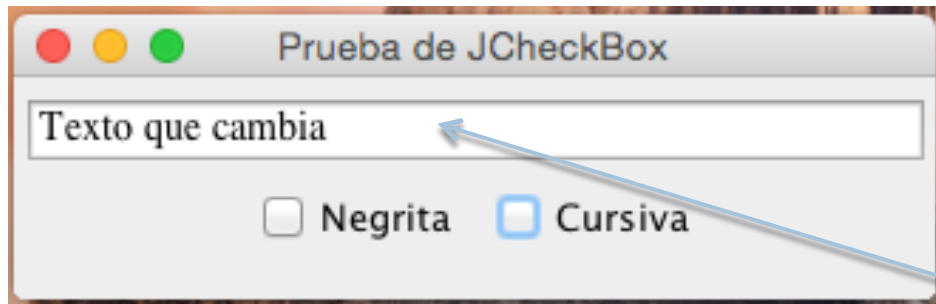
```
perroButton.addActionListener(myListener); ...
```

```
class RadioListener implements ActionListener ...
```

```
{ public void actionPerformed(ActionEvent e) {} } }
```



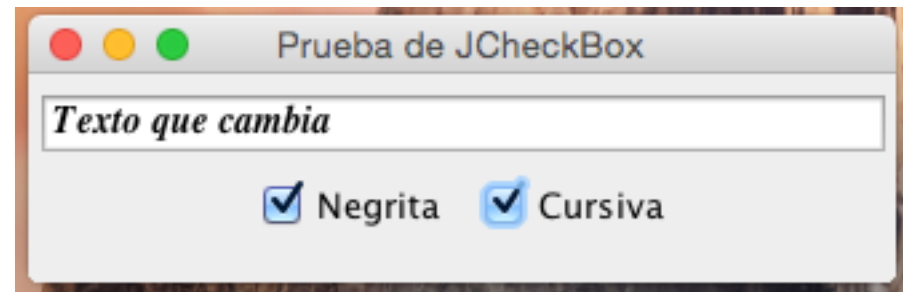
Ejercicio:



```
// Arial de 12 pts en negrita  
textField.setFont= new Font("Arial",Font.BOLD,12);
```

Las constantes de formato son:
Font.PLAIN, *Font.BOLD* y *Font.ITALIC*.

JTextField Muestra una línea de texto que puede ser editable. Con `setText("Texto")` se le asigna el texto.
`new JTextField("Texto que cambia", 28)`



```
□ public class PruebaCasillaVerificacion extends JFrame {
    private JTextField t;
    private JCheckBox negrita, cursiva;
    public PruebaCasillaVerificacion()
    {
        super( "Prueba de JCheckBox" );
        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        //creamos e inicializamos el objeto JTextField
        t = new JTextField( "Texto que cambia", 28 );
        //establecemos el estilo de la fuente y añadimos al panel
        t.setFont( new Font( "TimesRoman",Font.PLAIN, 14 ) );
        c.add( t );
        // crea los objetos casilla de verificación
        negrita = new JCheckBox( "Negrita" );
        c.add( negrita );
        cursiva = new JCheckBox( "Cursiva" );
        c.add( cursiva );
        //si se hace click en un objeto JCheckBox se genera un
        // ItemEvent que puede ser manejado por un ItemListener
        // (cualquier objeto de una clase que implemente la interfaz
        // ItemListener)
        //un objeto ItemListener debe definir el metodo
        // itemStateChanged
        ManejadorCasillaVerificacion manejador= new ManejadorCasillaVerificacion();
        negrita.addItemListener( manejador );
        cursiva.addItemListener( manejador );
    }
}
```



```
□ addWindowListener(  
    new WindowAdapter() {  
        public void windowClosing( WindowEvent e )  
        {  
            System.exit( 0 );  
        } // fin del método windowClosing  
    } // fin de la clase interna anónima  
); // fin de addWindowListener  
setSize( 325, 100 );  
setVisible(true);  
} // fin del constructor de PruebaCasillaVerificacion  
public static void main( String args[] )  
{  
    new PruebaCasillaVerificacion();  
}
```

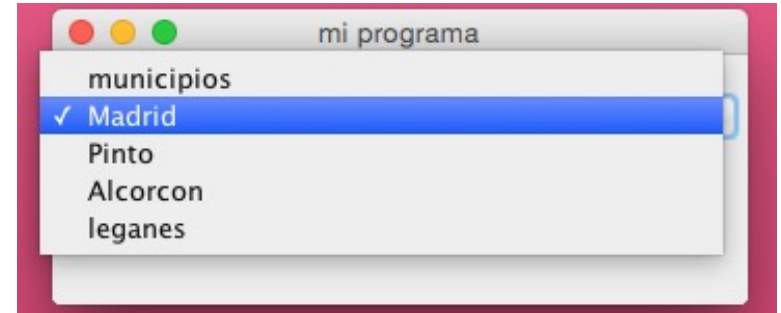
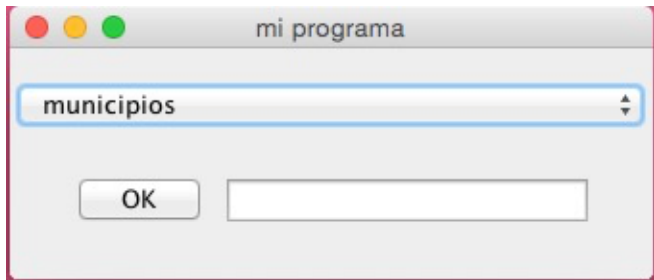
```

□ private class ManejadorCasillaVerificacion implements ItemListener {
    private int valNegrita = Font.PLAIN;
    private int valCursiva = Font.PLAIN;
    //con e.getSource() determinamos el onjeto sobre el que se hizo //click
    //con las estructuras if-else se determina cual fue modificado y //la accion que tenemos que llevar a cabo
    public void itemStateChanged( ItemEvent e )
    {
        if ( e.getSource() == negrita )
            if ( e.getStateChange() == ItemEvent.SELECTED )
                valNegrita = Font.BOLD;
            else
                valNegrita = Font.PLAIN;
        if ( e.getSource() == cursiva )
            if ( e.getStateChange() == ItemEvent.SELECTED )
                valCursiva = Font.ITALIC;
            else
                valCursiva = Font.PLAIN;
        //establecemos los tipos de la nueva fuente
        t.setFont(
            new Font( "TimesRoman", valNegrita + valCursiva, 14 ) );
    } // fin del método itemStateChanged
} // fin de la clase interna //ManejadorCasillaVerificacion
} // fin de la clase PruebaCasillaVerificacion

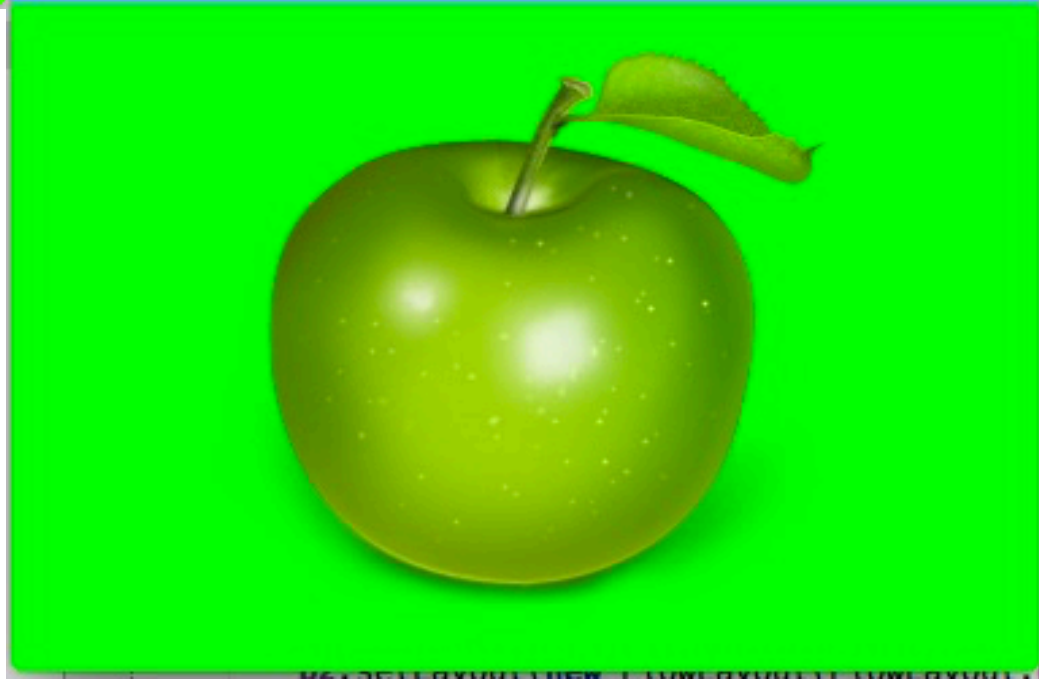
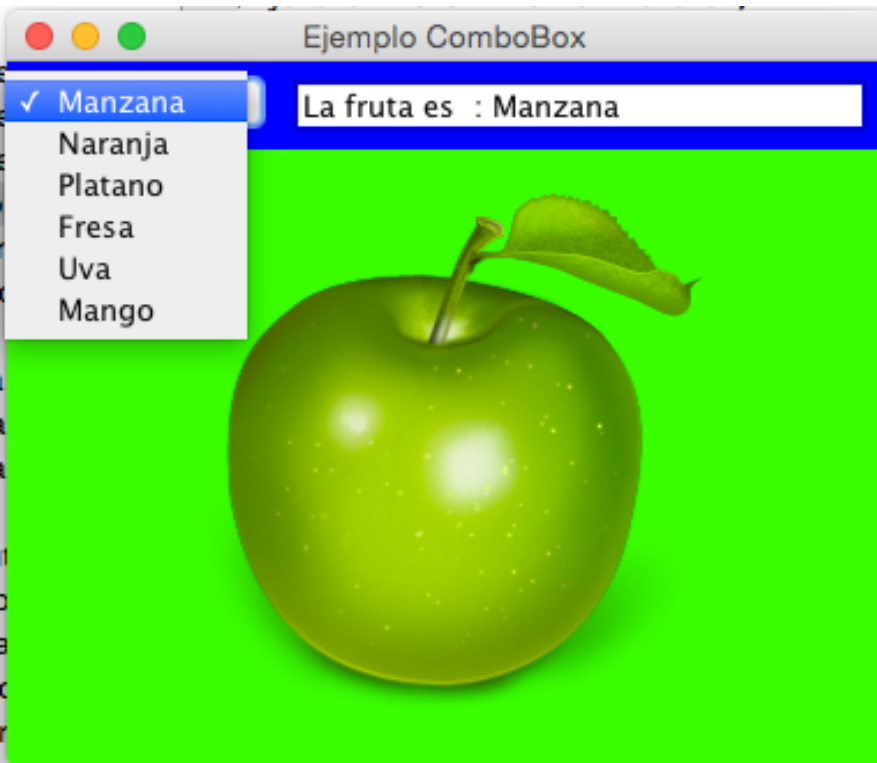
```

SWING Clase JComboBox

- Esta componente nos permite, al hacer click sobre ella, seleccionar una opción de entre un conjunto, todas ellas mutuamente exclusivas.



- Se pueden generalizar en dos tipos: Editables No editables
- Métodos:
 - ▣ void : addItem(Object item). Agrega un elemento al combo
 - ▣ Object: getSelectedItem(). Retorna el valor seleccionado.
 - ▣ int: getSelectedIndex(). Obtiene el índice seleccionado.
 - ▣ void: setEditable(boolean tipo). Determina si sólo mostrara los valores (false) o si se pueden escribir nuevos valores (true)



SWING Componentes para Texto

- JLabel Muestra texto, gráficos o ambos, sólo lectura. Ha de estar desactivado si lo está el componente.
- JTextField Muestra una línea de texto que puede ser editable. Con `setText("Texto")` se le asigna el texto.



JTextField : Miguel

- JPasswordField Oculta los caracteres introducidos por el usuario. `setEchoChar('char')` indica el carácter de máscara. `getPassword()` recupera el password introducido.



JPasswordField :

SWING Componentes para Texto

- **JFormattedTextField**. Permite introducir un campo de texto con formato, (si definimos que solo recibe números no permitirá letras, para esto se requiere definir la mascara a utilizar)

- `mascara= new MaskFormatter("#####");`



```
JFormattedTextField textField1 = new JFormattedTextField (new Integer(3));
```

```
// o bien ....
```

```
JFormattedTextField textField2 = new JFormattedTextField ();
```

```
textField2.setValue(new Integer(3));
```

```
Integer valor = textField1.getValue();
```

- Supongamos que el usuario escribe algo en el **JFormattedTextField** y luego pincha con el ratón en otro sitio (se dice que el **JFormattedTextField** pierde el foco). En el momento que el **JFormattedTextField** pierde el foco, comprueba el texto escrito por el usuario. Si es correcto, lo guarda de forma que el método **getValue()** nos devolverá el nuevo valor. Si es incorrecto, pondrá automáticamente el último valor bueno, deshaciendo el cambio hecho por el usuario.
- Este comportamiento puede cambiarse con el método **setFocusLostBehavior()**, al que podemos pasar varios valores:

SWING Componentes para Texto


- **JFormattedTextField.COMMIT.** Si el texto introducido es correcto, se guarda para devolverlo cuando se haga `getValue()`. Si es incorrecto, no se hace nada, el texto en pantalla queda como esta, o sea, mal. `getValue()` nos devolverá el último valor correcto, independientemente de lo que se muestre en pantalla.
- **JFormattedTextField.REVERT.** Cuando hacemos click en otro sitio, el editor vuelve automáticamente a su último valor bueno, descartando todas nuestras ediciones, sean correctas o no.
- **JFormattedTextField.COMMIT_OR_REVERT.** Esta es la opción por defecto y la más útil. Si el texto introducido es incorrecto, se vuelve automáticamente al último valor bueno conocido. Si el texto no es válido, se muestra el último valor bueno conocido.
- **JFormattedTextField.PERSIST.** Esta opción no hace nada con el texto introducido, independientemente de que esté bien o mal. `getValue()` siempre devolverá el último valor bueno conocido..

```
try { mascara= new MaskFormatter("#####");  
    cajaDeTextoConFormato= new JFormattedTextField(mascara);  
    //Importante definir el foco del componente, para que almacene el valor//  
    cajaDeTextoConFormato.setFocusLostBehavior(cajaDeTextoConFormato.COMMIT);}  
catch (ParseException e) { e.printStackTrace();}
```

Introduzca la Contraseña:


OK Ayuda

Mensaje

 Contacte con el administrador del Sistema Si ha olvidado la contraseña


Aceptar

Mensaje

 Enhorabuena! Ha escrito la contraseña correcta.

Aceptar

Mensaje de Error

 Contraseña Incorrecta. Intentelo de Nuevo.

Aceptar

Demo Caja de texto

Campo Formato	EEEEEEEEEEEEEE
Texto del Campo Formato	EEEEEEEEEEEEEE
contraseña	aaaaaaaaaaaaaaaaaaaaaa
contraseña en claro	232332323232323233

SWING Componentes para Texto

- JTextArea Espacio rectangular en el que ver y editar múltiples líneas de texto. Para que aparezcan barras de scroll debe ir dentro de un JScrollPane.

```
JTextArea: JTextField = Miguel  
JFormattedTextField = 111111  
JPasswordField = Contraseña  
JPasswordField Encriptado= [C@71369330  
check1 seleccionado
```

SWING Componentes para Texto

- JEditorPane , JTextPane permite vincular un área de texto con propiedades de formato, es decir, por ejemplo, podemos darle formato HTML a nuestro texto usando etiquetas, modificando el tamaño, color y hasta vinculando imágenes. JTextPane además permite otras opciones de formato, colores, iconos, trabajo con estilos, componentes entre otros

JEditorPane: **Texto En Negrilla y etiqueta H1**

Texto normal Texto con color azul fuente verdana

JTextPane: Texto con estilo rojo, seleccione un check ☒ check1 ☐ check2

Seleccione su edad

- ☐ 18-20
☒ 21-40
☐ 41-60
☐ 60 a más

Resultados

DNI 9999999999

Profesión

Ingeniero

[Imprimir Datos](#)

Edad entre : 21-40
de Profesión : Ingeniero
DNI numero:= 9999999999

```
areaEditorPane = new JEditorPane();
areaEditorPane.setBounds(90, 200, 520, 103);
/*Definimos el tipo de texto que utiliza*/
areaEditorPane.setContentType("text/html");
areaEditorPane.setText("<h1><b>Texto En Negrilla y etiqueta H1</b></h1>" +
    " Texto normal" +
    "<font color=\"blue\"> Texto con color azul</font>" +
    "<font face=\"verdana\"> fuente verdana</font>");

etiquetaJTextPane= new JLabel();
etiquetaJTextPane.setText("JTextPane: ");
etiquetaJTextPane.setBounds(20, 310, 100, 23);

/*usamos StyleContext para definir el estilo a usar*/
StyleContext estilo = new StyleContext();
/*creamos el estilo, no definimos nombre ni estilo padre...*/
Style estiloRojo = estilo.addStyle(null, null);
StyleConstants.setForeground( estiloRojo, Color.red );
/*definimos el estilo a usar*/
DefaultStyledDocument estiloPorDefecto = new DefaultStyledDocument( estilo );
areaTextPane = new JTextPane(estiloPorDefecto);
areaTextPane.setCharacterAttributes(estiloRojo, false);
areaTextPane.setBounds(90, 310, 520, 103);
```

// Se inserta

```
try {  
    areaTextPane.getStyledDocument().insertString(  
        areaTextPane.getStyledDocument().getLength(), "Texto con estilo rojo,  
seleccione un check ", estiloRojo);  
} catch (BadLocationException e) {  
    e.printStackTrace();  
}
```

```
/*Definimos el Foco del componente, para que se inserte de ultimo en el area*/  
areaTextPane.setCaretPosition(areaTextPane.getStyledDocument().getLength());  
check1 = new JCheckBox("check1");  
areaTextPane.insertComponent(check1);  
areaTextPane.setCaretPosition(areaTextPane.getStyledDocument().getLength());  
check2 = new JCheckBox("check2");  
areaTextPane.insertComponent(check2);
```

Ejercicio

JTextField : Miguel

JFormattedTextField : 111111

JPasswordField : ••••••••

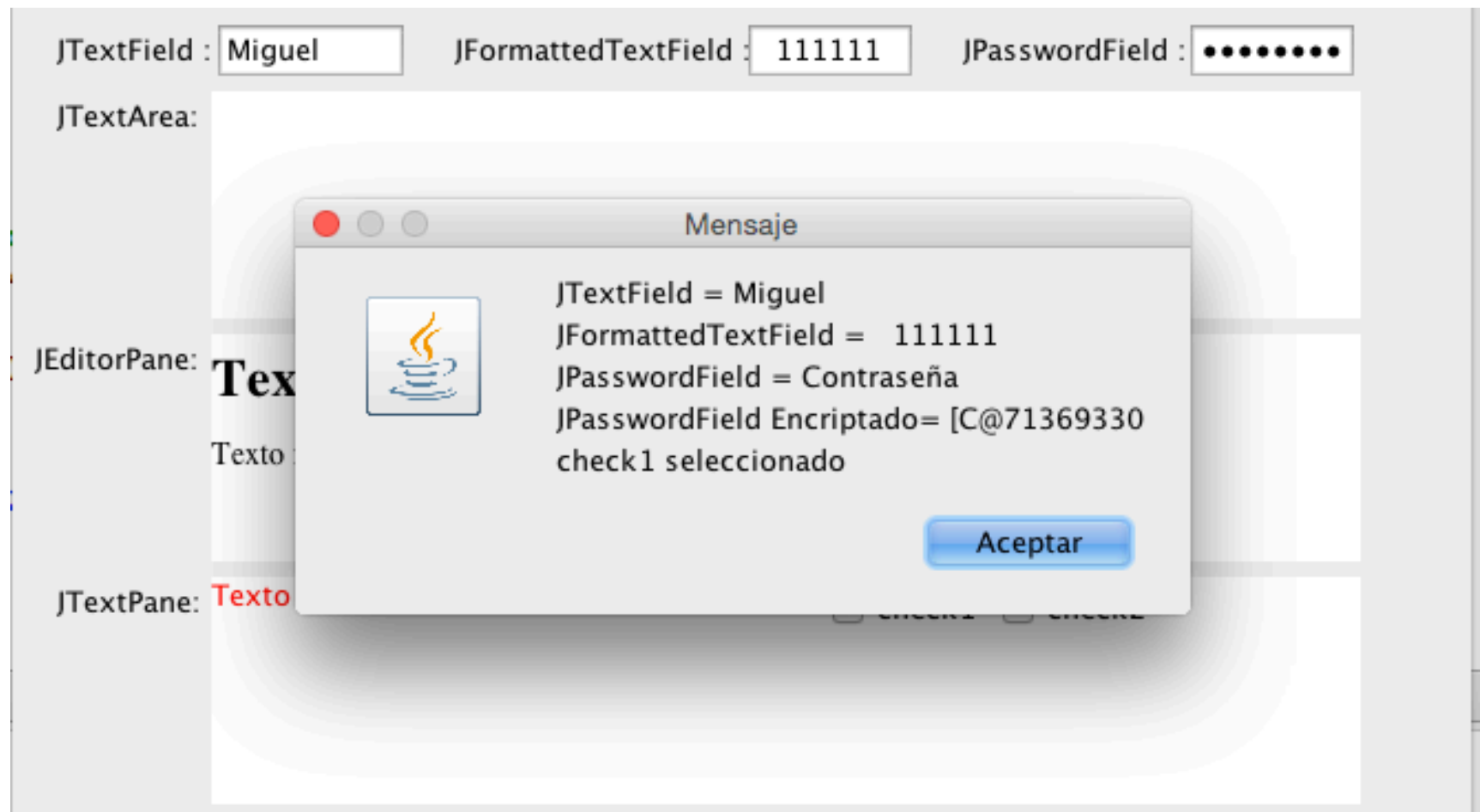
JTextArea:

JEditorPane: **Texto En Negrilla y etiqueta H1**

Texto normal Texto con color azul fuente verdana

JTextPane: Texto con estilo rojo, seleccione un check ☒ check1 ☐ check2

Ejercicio



Ejercicio

JTextField : Miguel

JFormattedTextField : 111111

JPasswordField : ●●●●●●●●

JTextArea: JTextField = Miguel
JFormattedTextField = 111111
JPasswordField = Contraseña
JPasswordField Encriptado= [C@71369330
check1 seleccionado

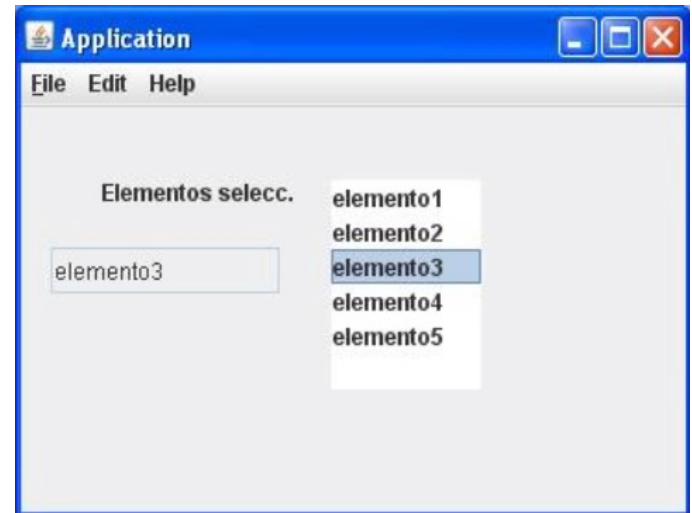
JEditorPane: **Texto En Negrilla y etiqueta H1**

Texto normal Texto con color azul fuente verdana

JTextPane: **Texto con estilo rojo, seleccione un check** ☒ check1 ☐ check2

SWING Clase JList

- Es un componente que muestra un conjunto de ítems de texto, gráfico o ambos.
- Permite tres tipos de selección:
 - ▣ Ítem único
 - ▣ Rango simple
 - ▣ Rango múltiple
- Mediante el método:
 - ▣ `.setSelectionMode(ListSelectionModel.SELECTION);`
 - Donde SELECTION puede ser:
 - `SINGLE_SELECTION` `SINGLE_INTERVAL_SELECTION`
 - `MULTIPLE_INTERVAL_SELECTION`



SWING Clase JList

- JList: Este componente nos permite presentar una lista de selección donde podemos escoger uno o varios elementos
 - ▣ **Agregar Elementos:**
 - 1: Para agregar elementos usando un arreglo es muy simple, tan solo tenemos que declarar nuestro array y agregárselo al constructor del objeto **JList** con el que estemos trabajando

```
JList listaNombres;  
String nombres[] = { "Cristian", "Julian", "Milena"};  
listaNombres = new JList( nombres );
```
 - 2: Tenemos que declarar un objeto de tipo **DefaultListModel** y por medio del método **addElement(elemento)**, vamos agregando elementos a nuestro modelo, posteriormente dicho modelo se agrega al **JList** con el que trabajemos:

```
JList listaNombres=new JList();  
DefaultListModel modelo = new DefaultListModel();  
modelo.addElement("Elemento1");  
modelo.addElement("Elemento2");  
modelo.addElement("Elemento3");  
listaNombres.setModel(modelo);
```

SWING Clase JList

□ Ejercicio JList: Paso para crear un Objeto JList:

```
//instanciamos la lista  
listaNombres = new JList();  
listaNombres.setSelectionMode(ListSelectionModel.SINGLE_SELECTION );
```

```
//instanciamos el modelo  
modelo = new DefaultListModel();
```

```
//instanciamos el Scroll que tendra la lista  
scrollLista = new JScrollPane();  
scrollLista.setBounds(20, 120,220, 80);  
scrollLista.setViewportViewView(listaNombres);
```

```
//los eventos de los botones  
public void actionPerformed(ActionEvent evento) {  
    if (evento.getSource()==agregar)  
    { agregarNombre();  
      mensaje.setText("Se agregó un nuevo elemento");}  
    if (evento.getSource()==eliminar)  
    {eliminarNombre(listaNombres.getSelectedIndex() ); }  
    if (evento.getSource()==borrar){  
      borrarLista();  
      mensaje.setText("Se borró toda la lista"); }}
```

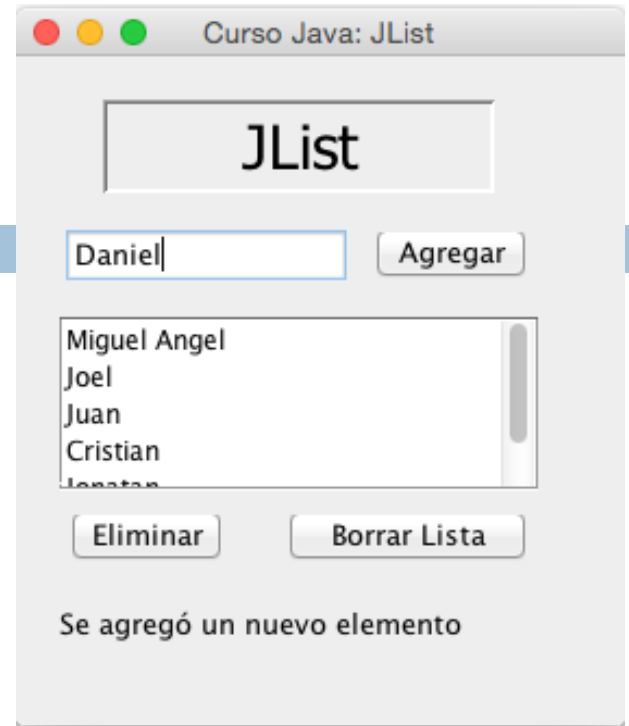


SWING Clase JList

```
private void agregarNombre() {  
    String nombre=campo.getText();  
    modelo.addElement(nombre);  
    listaNombres.setModel(modelo);  
    campo.setText("");}
```

```
private void eliminarNombre(int indice) {  
    if (indice>=0) {  
        modelo.removeElementAt(indice);  
        mensaje.setText("Se eliminó un elemento en la posición "+indice);  
    }else{  
        JOptionPane.showMessageDialog(null, "Debe seleccionar un indice"  
        ,"Error", JOptionPane.ERROR_MESSAGE);  
        mensaje.setText("NO se seleccionó ningún elemento");}}
```

```
private void borrarLista() {  
    modelo.clear();}
```



JList

- Instanciamos un JList
- **listaNombres = new JList();**
listaNombres.setSelectionMode(ListSelectionMode.SINGLE_SELECTION);
definimos un modelo
- **private DefaultListModel modelo;***/*declaramos el Modelo*/*
- *//instanciamos el modelo*
modelo = new DefaultListModel();
listaNombres.setModel(modelo);
- Definimos un JScrollPane
- **private JScrollPane scrollLista;**
- *//instanciamos el Scroll que tendra la lista*
scrollLista = new JScrollPane();
scrollLista.setBounds(20, 120,220, 80);
scrollLista.setViewportViewView(listaNombres);
- Añadimos el JScrollPane al panel
- **panel2.add(scrollLista,BorderLayout.NORTH**
- **Los elementos se introducen en el modelo**
 - **modelo.addElement(nombre);**

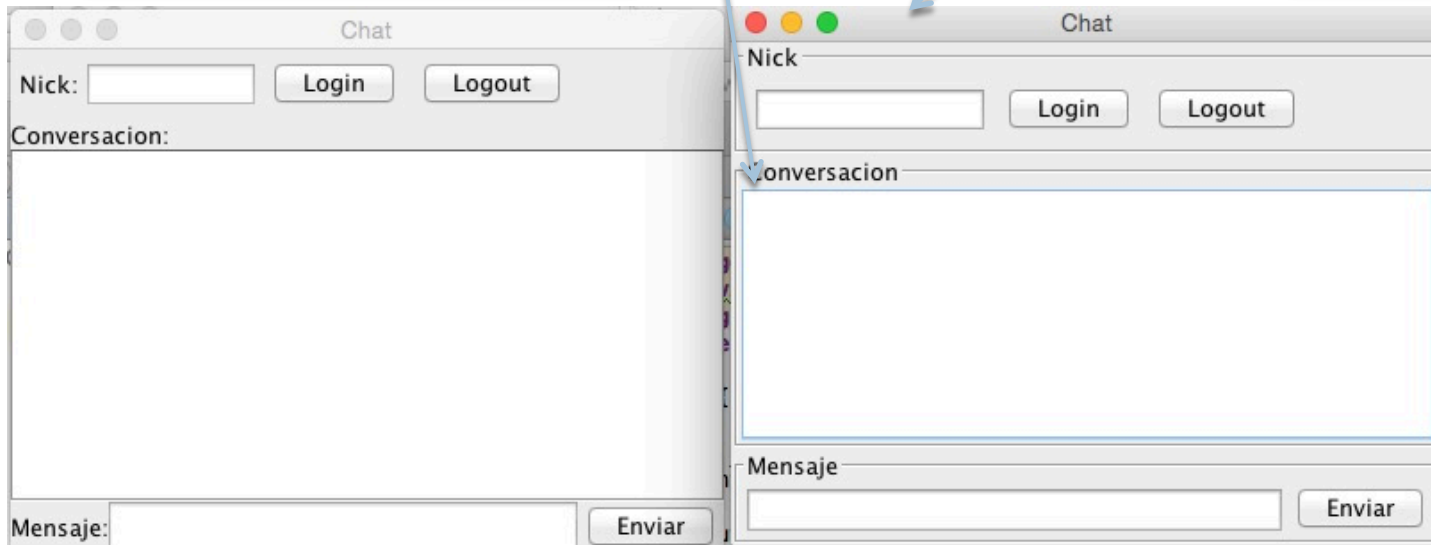
SWING

Veamos como quedaría el ejemplo del Chat con *Swing*.

En la región central, hay un JPanel con un BorderLayout en cuya región central hay una JList.

Por último, en la región sur hay un JPanel con un BorderLayout que tiene un JTextField en la región central y un JButton en la región este.

La ventana principal (una instancia de JFrame) tiene un BorderLayout. En la región norte, hay un JPanel con un FlowLayout alineado a izquierda con tres componentes: un JTextField y dos JButton.



SWING

En Swing no trabajamos sobre el JFrame sino que lo hacemos sobre un panel contenedor que se llama *content* y que lo obtenemos con el método `getContentPane()`.

El *layout* de la ventana principal lo definimos sobre este *content*. Los componentes que contendrá la ventana principal también los agregamos sobre *content*

El código del chat empleando Swing sería el siguiente:

```
import java.awt.*;
import javax.swing.*; import javax.swing.border.*;
public class ChatSwing extends JFrame {
    private JTextField tfNick;
    private JTextField tfMensaje;
    private JButton bLogin;
    private JButton bLogout;
    private JButton bEnviar;
    private JList lstLog;
    private Border border;
    public ChatSwing() {
        super("Chat");
        Container content = getContentPane();
        content.setLayout(new BorderLayout());
        border = BorderFactory.createEtchedBorder(EtchedBorder.LOWERED);
        JPanel pNorth = _crearPNorte();
        content.add(pNorth, BorderLayout.NORTH);
        JPanel pCenter = _crearPCenter();
        content.add(pCenter, BorderLayout.CENTER);
        JPanel pSouth = _crearPSur();
        content.add(pSouth, BorderLayout.SOUTH);
        setSize(400, 300);
        setVisible(true);
    }
}
```

pedimos el "panel contenedor"
al JFrame

configuramos el layout

este será el tipo de borde que
utilizamos en todos los paneles

creamos el panel norte,
central y sur

SWING

Con respecto de la versión de AWT instanciamos un `TitledBorder` (a través de la clase `BorderFactory`) para asignarlo al `JPanel`.

```
private JPanel _crearPNorte() {  
    JPanel p = new JPanel(new FlowLayout(FlowLayout.LEFT));  
    // que cree en el constructor  
    TitledBorder titulo = BorderFactory.createTitledBorder(border, "Nick");  
    p.setBorder(titulo);  
    tfNick = new JTextField(10);  
    p.add(tfNick);  
    bLogin = new JButton("Login");  
    p.add(bLogin);  
    bLogout = new JButton("Logout");  
    p.add(bLogout);  
    return p;  
}
```

instancio un `TitledBorder` con el titulo y el objeto border creado en el constructor

añado el `JtextField` con el metodo `add` del `Jpanel`

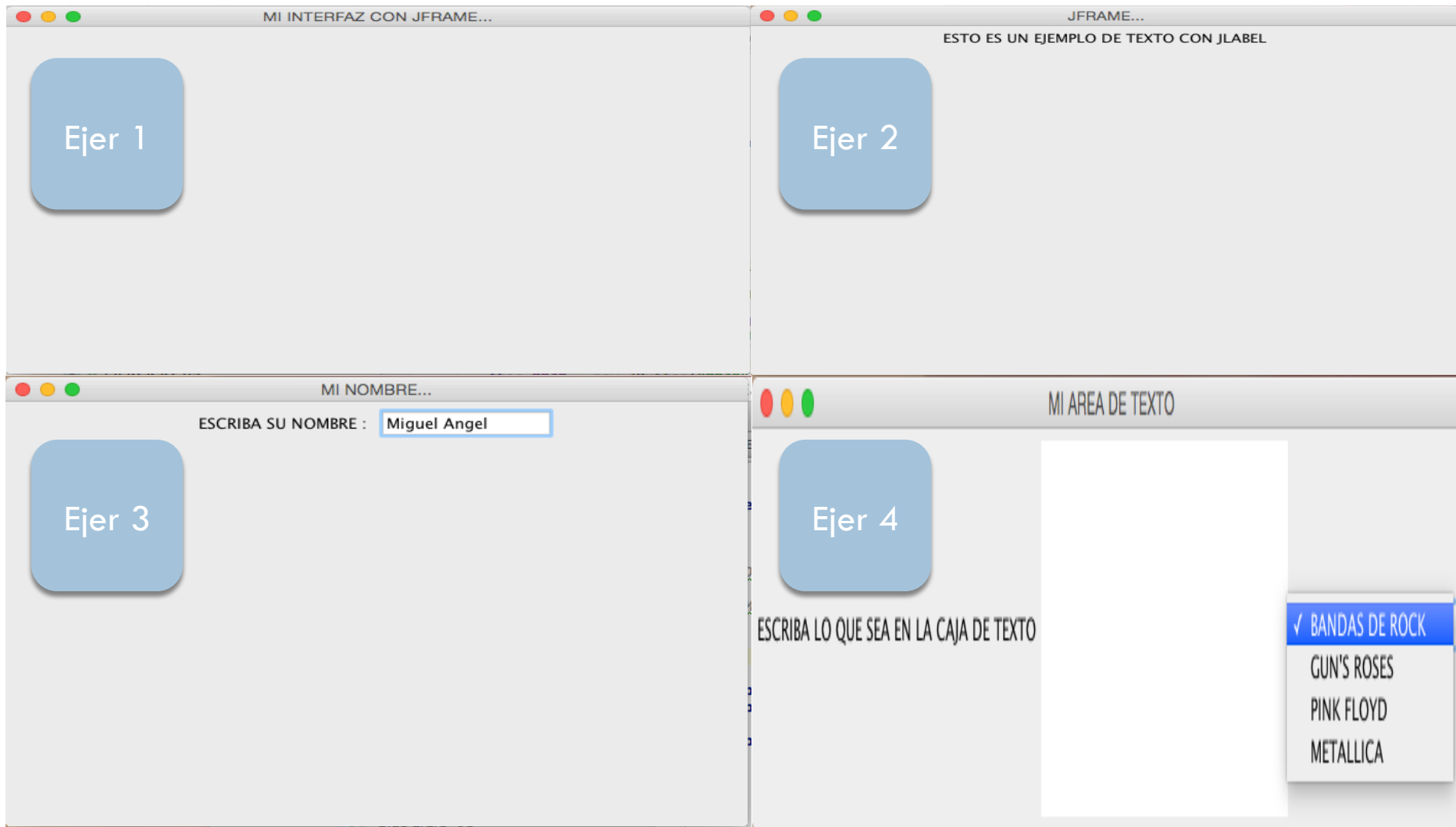
```
private JPanel _crearPCenter() {  
    JPanel p = new JPanel(new BorderLayout());  
    TitledBorder titulo = BorderFactory.createTitledBorder(border, "Conversacion");  
    p.setBorder(titulo);  
    lstLog = new JList();  
    JScrollPane scroll = new JScrollPane(lstLog);  
    p.add(scroll, BorderLayout.CENTER);  
    return p;  
}
```


SWING

```
private JPanel _crearPSur() {
    JPanel p = new JPanel(new BorderLayout());
    TitledBorder titulo = BorderFactory.createTitledBorder(border, "Mensaje");
    p.setBorder(titulo);
    tfMensaje = new JTextField();
    p.add(tfMensaje, BorderLayout.CENTER);
    bEnviar = new JButton("Enviar");
    p.add(bEnviar, BorderLayout.EAST);
    return p;
}

public static void main(String args[]) throws Exception {
    ChatSwing c = new ChatSwing();
}
}
```

SWING



Ejercicio 1

```
import javax.swing.*;

public class Ejer1 extends JFrame
{
    //Le agregamos todo a la ventana mediante el constructor

    public Ejer1 ()
    {
        // Configuramos la ventana

        super("MI INTERFAZ CON JFRAME..."); // Le ponemos un titulo
        this.setSize(600,400); //-- Le damos un tamaño a la ventana
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Hacemos que se cierre por defecto
    }

    public static void main(String[] ARGS)
    {
        //Hacemos que se inicialize nuestra ventana JFrame

        Ejer1 miGui = new Ejer1();

        //Hacemos que nuestra ventana se visible

        miGui.setVisible(true);
    }
}
```

Ejercicio2

```
package mad2;
import javax.swing.*;
import java.awt.*;
public class Ejer2 extends JFrame
{ public JLabel BIENVENIDA;
  public Ejer2()
  {/-- CONFIGURAMOS LA VENTANA
    super(" JFRAME..."); //-- LE PONEMOS UN TITULO
    this.setSize(600,400); //-- LE DAMOS UN TAMAÑO A LA VENTANA
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Hacemos que se cierre por defecto
    FlowLayout DISTRIBUCION = new FlowLayout();//Le Agregamos un layout
    this.setLayout(DISTRIBUCION);
    this.BIENVENIDA = new JLabel("ESTO ES UN EJEMPLO DE TEXTO CON JLABEL");
    this.add(this.BIENVENIDA);
  }
  public static void main(String[] ARGS)
  {/-- HACEMOS QUE SE INICIALIZE NUESTRA VENTANA JFRAME
    Ejer2 miGui = new Ejer2();
    miGui.setVisible(true); //-- HACEMOS QUE NUESTRA VENTANA SE VISIBLE
  }
}
```

Ejercicio3

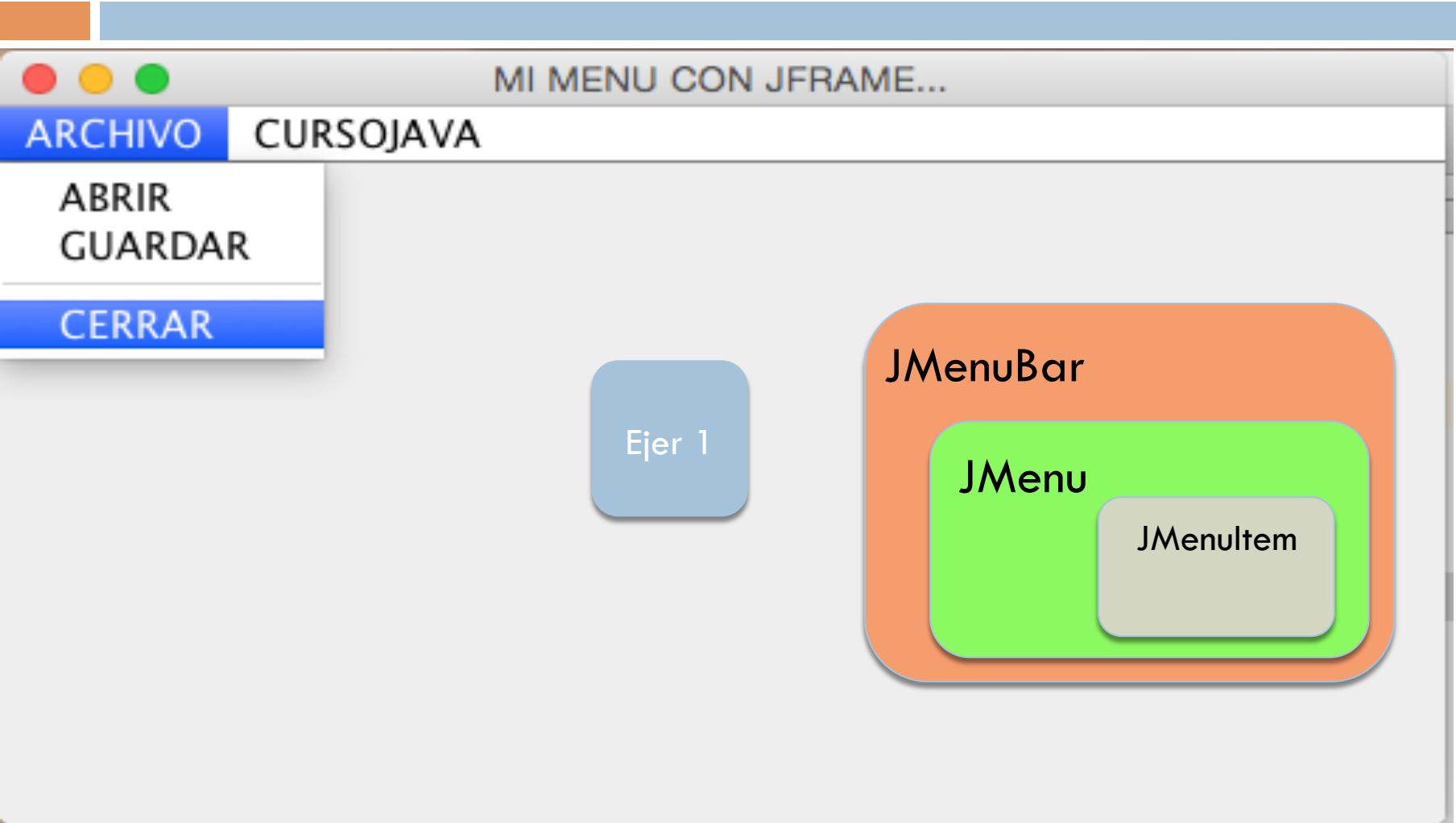
```
import javax.swing.*; import java.awt.*;
public class Ejer3 extends JFrame
{public JLabel INDIQUE_NOMBRE;
public JTextField MI_NOMBRE;
public Ejer3()
{ super("MI NOMBRE...");//-- CONFIGURAMOS LA VENTANA
this.setSize(600,400);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
FlowLayout DISTRIBUIDOR = new FlowLayout();//-- LE AGREGAMOS UN DISTRIBUIDOR DE COMPONENTES
this.setLayout(DISTRIBUIDOR);
//-- AGREGAMOS TODOS LOS COMPONENTES
this.INDIQUE_NOMBRE = new JLabel("ESCRIBA SU NOMBRE : ");
this.add(this.INDIQUE_NOMBRE);
this.MI_NOMBRE = new JTextField("SU NOMBRE AQUI...");
this.add(this.MI_NOMBRE); }
public static void main(String[] ARGS)
{/-- HACEMOS QUE SE INICIALIZE NUESTRA VENTANA JFrame
Ejer3 miGui = new Ejer3();
/-- HACEMOS QUE NUESTRA VENTANA SE VISIBLE
miGui.setVisible(true);
}
}
```

Ejercicio4

```
import javax.swing.*; import java.awt.*;
public class Ejer4 extends JFrame
{ public JTextArea MI_CAJA_TEXTO; public JLabel MI_TEXTO; public JComboBox LISTA_DESPLEGABLE;
  public Ejer5()
  { super("MI AREA DE TEXTO");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    FlowLayout DISTRIBUCION = new FlowLayout();
    this.setLayout(DISTRIBUCION);
    this.MI_TEXTO = new JLabel();
    this.MI_TEXTO.setText("ESCRIBA LO QUE SEA EN LA CAJA DE TEXTO");
    this.add(this.MI_TEXTO);
    this.MI_CAJA_TEXTO = new JTextArea(10,20);
    this.add(MI_CAJA_TEXTO);
    this.LISTA_DESPLEGABLE = new JComboBox();
    this.LISTA_DESPLEGABLE.addItem("BANDAS DE ROCK");
    this.LISTA_DESPLEGABLE.addItem("GUN'S ROSES");
    this.LISTA_DESPLEGABLE.addItem("PINK FLOYD");
    this.LISTA_DESPLEGABLE.addItem("METALLICA");
    this.add(this.LISTA_DESPLEGABLE); }
  public static void main(String[] ARGS)
  { Ejer5 miGui = new Ejer5();
    miGui.pack();
    miGui.setVisible(true);
  }
}
```

EL METODO PACK SIRVE PARA DAR TAMAÑO A TODOS LOS COMPONENTES POR DEFECTO

SWING Menus



Ejercicio 1

```
import javax.swing.*;

public class Ejer1 extends JFrame
{
    public JMenu MENU_ARCHIVO,MENU_JAVA;
    public JMenuItem ITEM_ABRIR;
    public JMenuItem ITEM_GUARDAR;
    public JMenuItem ITEM_CERRAR;
    public JMenuItem ITEM_JAVADHC;
    public JMenuItem ITEM_MAD1;
    public JMenuItem ITEM_MAD2;
    public JMenuItem ITEM_MAD3;
    public JMenuBar MENU_BAR;
    public Ejer1()
    {
        super("MI MENU CON JFRAME...");
        this.setSize(1200,800);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.ITEM_ABRIR = new JMenuItem("ABRIR");
        this.ITEM_GUARDAR = new JMenuItem("GUARDAR");
        this.ITEM_CERRAR = new JMenuItem("CERRAR");
        this.ITEM_JAVADHC = new JMenuItem("CURSOJAVA");
        this.ITEM_MAD1 = new JMenuItem("MAD1");
        this.ITEM_MAD2 = new JMenuItem("MAD2");
        this.ITEM_MAD3 = new JMenuItem("MAD3");
    }
}
```


Ejercicio 1

```
this.MENU_ARCHIVO = new JMenu("ARCHIVO");  
this.MENU_ARCHIVO.add(this.ITEM_ABRIR);  
this.MENU_ARCHIVO.add(this.ITEM_GUARDAR);  
this.MENU_ARCHIVO.addSeparator();  
this.MENU_ARCHIVO.add(this.ITEM_CERRAR);  
this.MENU_JAVA = new JMenu("CURSOJAVA");  
this.MENU_JAVA.add(this.ITEM_JAVADHC);  
this.MENU_JAVA.add(this.ITEM_MAD1);  
this.MENU_JAVA.addSeparator();  
this.MENU_JAVA.add(this.ITEM_MAD2);  
this.MENU_JAVA.add(this.ITEM_MAD3);  
  
this.MENU_BAR = new JMenuBar();  
this.MENU_BAR.add(this.MENU_ARCHIVO);  
this.MENU_BAR.add(this.MENU_JAVA);  
  
this.setJMenuBar(this.MENU_BAR);  
}  
  
public static void main(String[] ARGs)  
{  
    Ejer1 miGui = new Ejer1();  
  
    miGui.setVisible(true);  
}  
}
```

SWING Menus



Ejercicio 2

```
import javax.swing.*;
import java.awt.*;

public class Ejer2 extends JFrame
{
    public JButton BTN_MUSICA,BTN_VIDEO,BTN_INTERNET,BTN_JAVADHC,BTN_INFO;
    public JToolBar CAJA_HERRAMIENTA_01,CAJA_HERRAMIENTA_02;
    public ImagenIcon IMG_MUSICA,IMG_VIDEO,IMG_INTERNET,IMG_JAVADHC,IMG_INFO;
    public Ejer2()
    {
        super("MI MENU CON JFRAME...");
        this.setSize(600,400);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.IMG_VIDEO = new ImagenIcon("IMAGENES/VIDEO.PNG");
        this.IMG_MUSICA = new ImagenIcon("IMAGENES/MUSICA.PNG");
        this.IMG_INTERNET = new ImagenIcon("IMAGENES/INTERNET.PNG");
        this.IMG_JAVADHC = new ImagenIcon("IMAGENES/JAVADHC.PNG");
        this.IMG_INFO = new ImagenIcon("IMAGENES/INFO1.PNG");
        this.BTN_VIDEO = new JButton("VIDEO",this.IMG_VIDEO);
        this.BTN_MUSICA = new JButton("MUSICA",this.IMG_MUSICA);
        this.BTN_INTERNET = new JButton("INTERNET",this.IMG_INTERNET);
        this.BTN_JAVADHC = new JButton("CURSOJAVA",this.IMG_JAVADHC);
        this.BTN_INFO = new JButton("Este es el curso de ANDROID",this.IMG_INFO);
    }
}
```

Ejercicio 2

```
this.CAJA_HERRAMIENTA_01 = new JToolBar();  
this.CAJA_HERRAMIENTA_01.add(this.BTN_VIDEO);  
this.CAJA_HERRAMIENTA_01.add(this.BTN_MUSICA);  
this.CAJA_HERRAMIENTA_01.add(this.BTN_INTERNET);
```

```
this.CAJA_HERRAMIENTA_02 = new JToolBar();  
this.CAJA_HERRAMIENTA_02.add(this.BTN_JAVADHC);  
this.CAJA_HERRAMIENTA_02.add(this.BTN_INFO);
```

```
BorderLayout DISTRIBUIDOR = new BorderLayout();  
this.setLayout(DISTRIBUIDOR);
```

```
this.add(this.CAJA_HERRAMIENTA_01, BorderLayout.NORTH);  
this.add(this.CAJA_HERRAMIENTA_02, BorderLayout.SOUTH);
```

```
}
```

```
public static void main(String[] ARGUMENTOS)
```

```
{
```

```
    Ejer2 miGui = new Ejer2();
```

```
    //miGui.pack();
```

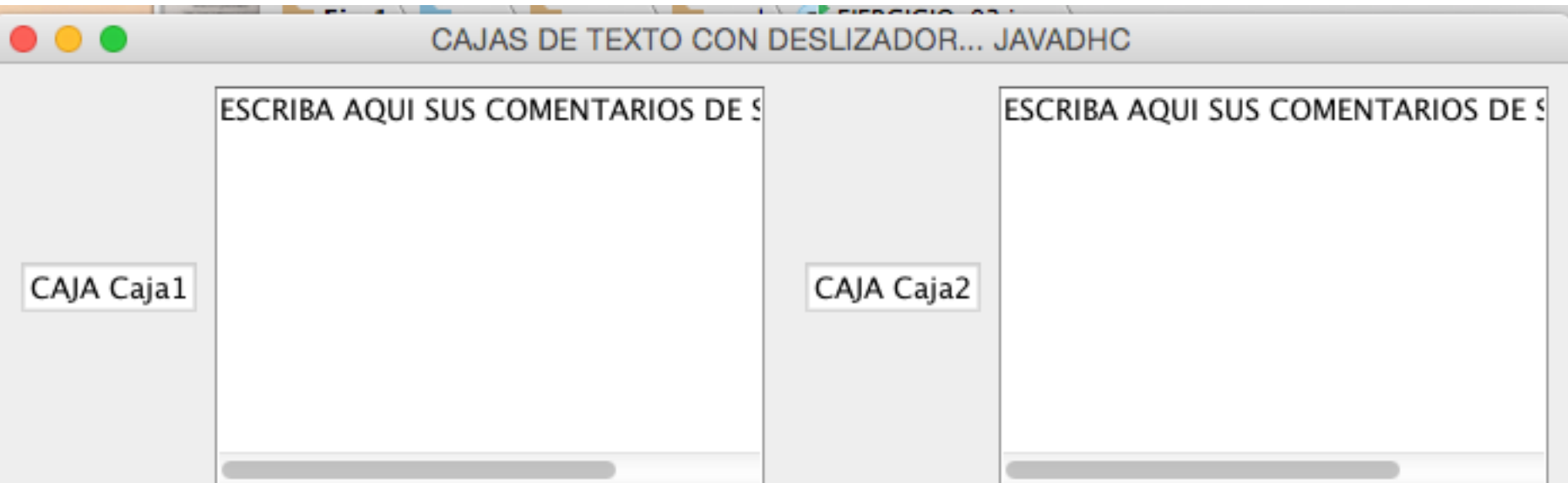
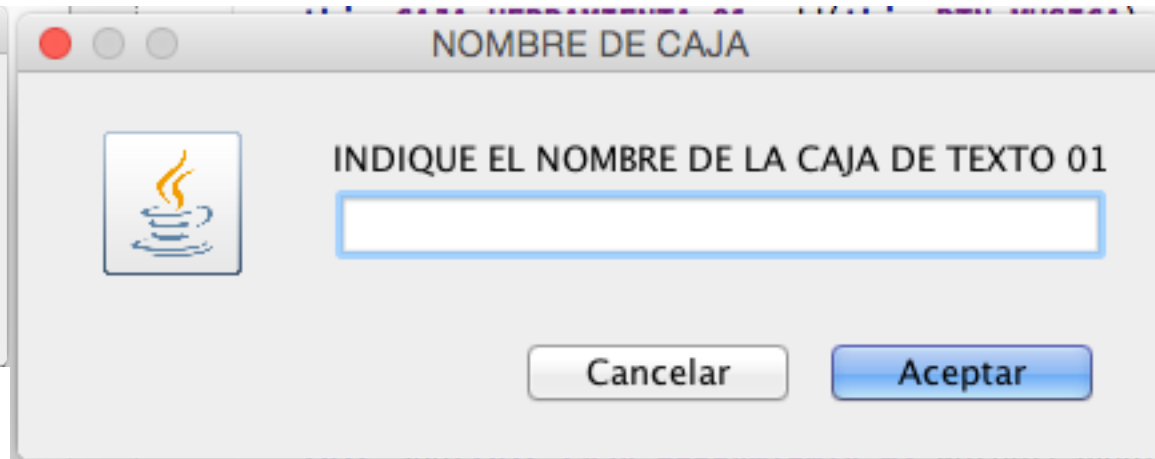
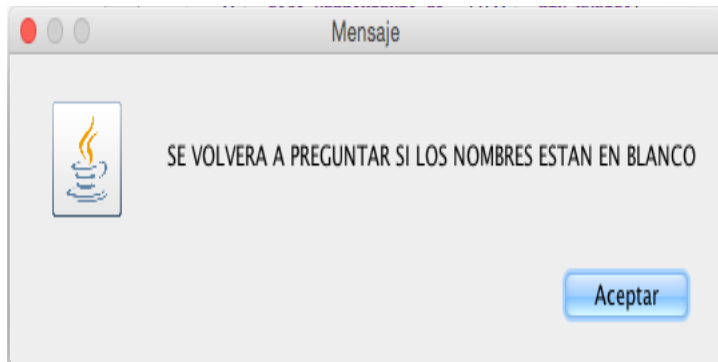
```
    miGui.setVisible(true);
```

```
}
```

```
}
```

SWING Menus

Ejer 3



Ejercicio 3

```
import javax.swing.*; import java.awt.*;
public class Ejer3 extends JFrame
{
    public MI_PANEL PANEL_01, PANEL_02;
    public JScrollPane DESLIZADOR;
    public Ejer3()
    {
        super("CAJAS DE TEXTO CON DESLIZADOR... JAVADHC");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        String NOMBRE_CAJA_01, NOMBRE_CAJA_02;
        do {JOptionPane.showMessageDialog(null,"SE VOLVERA A PREGUNTAR SI LOS NOMBRES ESTAN
EN BLANCO");
            NOMBRE_CAJA_01 = JOptionPane.showInputDialog(null,"INDIQUE EL NOMBRE DE LA CAJA
DE TEXTO 01","NOMBRE DE CAJA",JOptionPane.INFORMATION_MESSAGE);
            NOMBRE_CAJA_02 = JOptionPane.showInputDialog(null,"INDIQUE EL NOMBRE DE LA CAJA
DE TEXTO 02","NOMBRE DE CAJA",JOptionPane.INFORMATION_MESSAGE); }
        while(NOMBRE_CAJA_01.isEmpty() || NOMBRE_CAJA_02.isEmpty());
        FlowLayout DISTRIBUIDOR = new FlowLayout();
        this.setLayout(DISTRIBUIDOR);
        PANEL_01 = new MI_PANEL(NOMBRE_CAJA_01);
        PANEL_02 = new MI_PANEL(NOMBRE_CAJA_02);
        this.add(PANEL_01);
        this.add(PANEL_02);}
    public static void main(String[] ARGUMENTOS)
    {
        Ejer3 miGui = new Ejer3();
        miGui.pack();
        miGui.setVisible(true);
    }
}
```

Ejercicio 3

```
class MI_PANEL extends JPanel
{
    public JScrollPane DESLIZADOR;
    public JTextArea COMENTARIOS;
    public JTextField NOMBRE_CAJA;

    public MI_PANEL(String NOMBRE)
    {
        super();

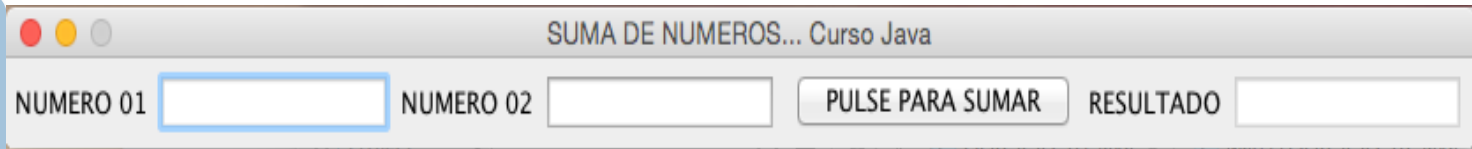
        this.NOMBRE_CAJA = new JTextField();
        this.NOMBRE_CAJA.setText("CAJA " + NOMBRE);
        this.NOMBRE_CAJA.setEditable(false);
        this.add(this.NOMBRE_CAJA);

        this.COMENTARIOS = new JTextArea(10,20);
        this.COMENTARIOS.setText("ESCRIBA AQUI SUS COMENTARIOS DE SU CAJA " + NOMBRE);

        this.DESLIZADOR = new JScrollPane(this.COMENTARIOS);
        this.add(this.DESLIZADOR);
    }
}
```

SWING Menus

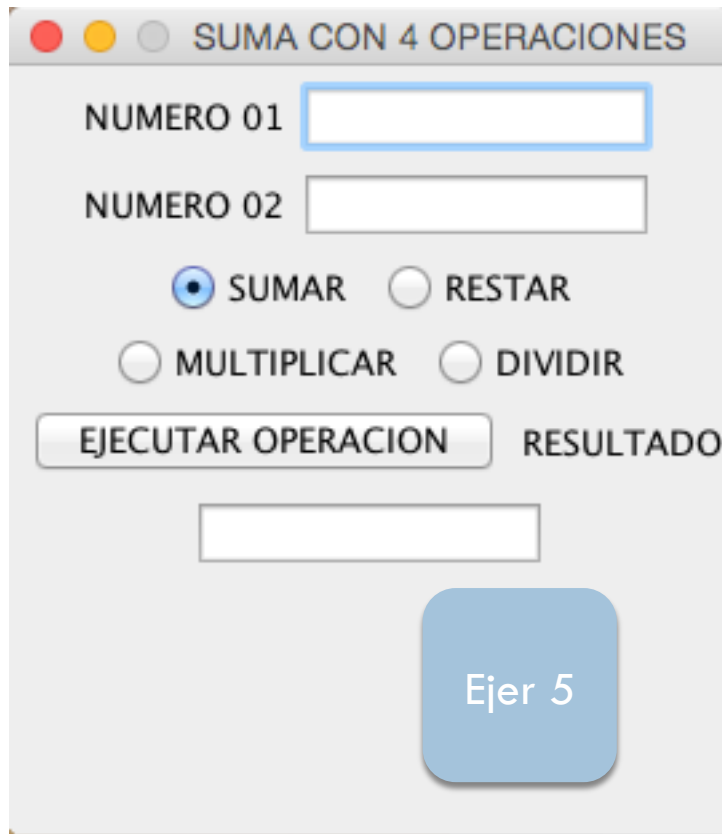
Ejer 4



SUMA DE NUMEROS... Curso Java

NUMERO 01 NUMERO 02

RESULTADO



SUMA CON 4 OPERACIONES

NUMERO 01

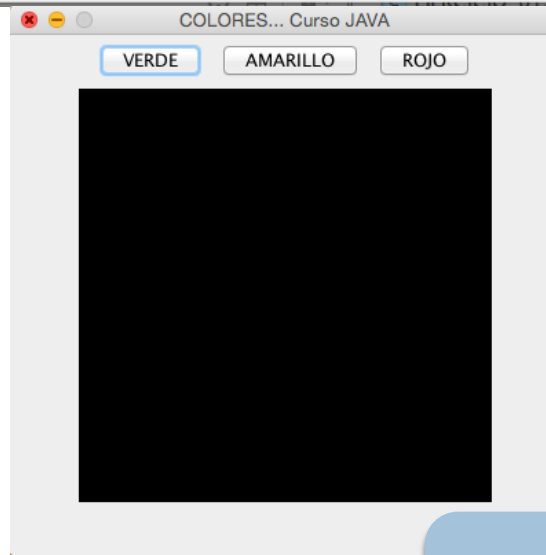
NUMERO 02

☒ SUMAR ☐ RESTAR

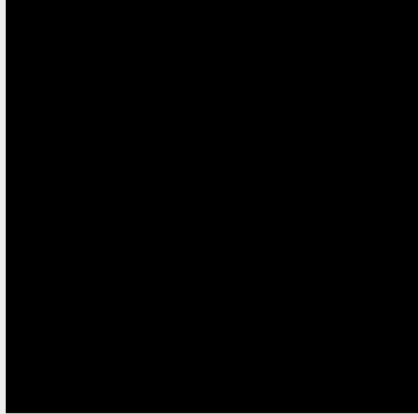
☐ MULTIPLICAR ☐ DIVIDIR

RESULTADO

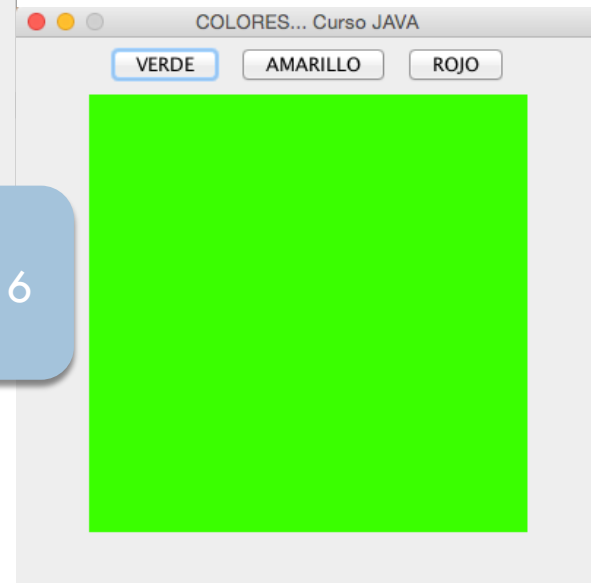
Ejer 5



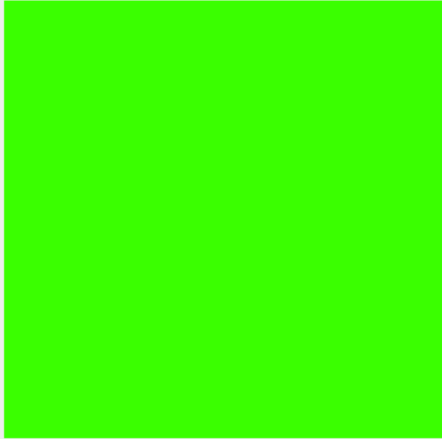
COLORES... Curso JAVA



Ejer 6



COLORES... Curso JAVA



Ejercicio 4

```
import java.awt.*;import java.awt.event.*;import javax.swing.*;
public class Ejer4 extends JFrame implements ActionListener
{
    public JLabel LETRA_NUMERO_01,LETRA_NUMERO_02,LETRA_RESULTADO;
    public JTextField NUMERO_01,NUMERO_02,RESULTADO;
    public JButton SUMAR;
    public Ejer4()
    {
        super("SUMA DE NUMEROS... Curso Java");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(310,460);
        this.LETRA_NUMERO_01 = new JLabel("NUMERO 01");
        this.LETRA_NUMERO_02 = new JLabel("NUMERO 02");
        this.LETRA_RESULTADO = new JLabel("RESULTADO");
        this.NUMERO_01 = new JTextField(10);
        this.NUMERO_02 = new JTextField(10);
        this.RESULTADO = new JTextField(10);
        this.RESULTADO.setEditable(false);
        this.SUMAR = new JButton("PULSE PARA SUMAR");
        this.SUMAR.addActionListener(this);
        FlowLayout DISTRIBUIDOR = new FlowLayout();
        this.setLayout(DISTRIBUIDOR);
        this.add(this.LETRA_NUMERO_01);
        this.add(this.NUMERO_01);
        this.add(this.LETRA_NUMERO_02);
        this.add(this.NUMERO_02);
        this.add(this.SUMAR);
        this.add(this.LETRA_RESULTADO);
        this.add(this.RESULTADO);
    }
}
```

Ejercicio 4

```
public void actionPerformed(ActionEvent EVENTO)
{
    float AUX_NUMERO_01,AUX_NUMERO_02,AUX_RESULTADO;
    try
    {
        AUX_NUMERO_01 = Float.parseFloat(this.NUMERO_01.getText());
        AUX_NUMERO_02 = Float.parseFloat(this.NUMERO_02.getText());
        AUX_RESULTADO = AUX_NUMERO_01 + AUX_NUMERO_02;
        this.RESULTADO.setText(String.valueOf(AUX_RESULTADO));
    }
    catch(Exception E)
    {
        this.RESULTADO.setText("ERROR AL SUMAR");
    }
}

public static void main(String[] ARGUMENTOS)
{
    Ejer4 miGui = new Ejer4();
    miGui.setResizable(false);

    miGui.setVisible(true);
}
}
```

Ejercicio 5

```
public class Ejer5 extends JFrame implements ActionListener
{
    JRadioButton SUMAR, RESTAR, MULTIPLICAR, DIVIDIR;
    ButtonGroup OPERACIONES;
    JButton EJECUTAR_OPERACION;
    JLabel LETRA_NUMERO_01, LETRA_NUMERO_02, LETRA_RESULTADO;
    JTextField NUMERO_01, NUMERO_02, RESULTADO;

    public Ejer5()
    {
        super("SUMA CON 4 OPERACIONES");
        this.setSize(270, 310);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.LETRA_NUMERO_01 = new JLabel("NUMERO 01");
        this.LETRA_NUMERO_02 = new JLabel("NUMERO 02");
        this.LETRA_RESULTADO = new JLabel("RESULTADO");
        this.NUMERO_01 = new JTextField(10);
        this.NUMERO_02 = new JTextField(10);
        this.RESULTADO = new JTextField(10);
        this.EJECUTAR_OPERACION = new JButton("EJECUTAR OPERACION");
        this.EJECUTAR_OPERACION.addActionListener(this);
    }
}
```

Ejercicio 5

```
this.SUMAR = new JRadioButton("SUMAR", true);
this.RESTAR = new JRadioButton("RESTAR", false);
this.MULTIPLICAR = new JRadioButton("MULTIPLICAR", false);
this.DIVIDIR = new JRadioButton("DIVIDIR", false);
this.OPERACIONES = new ButtonGroup();
this.OPERACIONES.add(this.SUMAR);
this.OPERACIONES.add(this.RESTAR);
this.OPERACIONES.add(this.MULTIPLICAR);
this.OPERACIONES.add(this.DIVIDIR);
FlowLayout DISTRIBUIDOR = new FlowLayout();
this.setLayout(DISTRIBUIDOR);
this.add(this.LETRA_NUMERO_01);
this.add(this.NUMERO_01);
this.add(this.LETRA_NUMERO_02);
this.add(this.NUMERO_02);
this.add(this.SUMAR);
this.add(this.RESTAR);
this.add(this.MULTIPLICAR);
this.add(this.DIVIDIR);
this.add(this.EJECUTAR_OPERACION);
this.add(this.LETRA_RESULTADO);
this.add(this.RESULTADO);
```

```
}
```

Ejercicio 5

```
public void actionPerformed(ActionEvent EVENTO)
{
    float AUX_NUMERO_01, AUX_NUMERO_02, AUX_RESULTADO;
    try {
        AUX_RESULTADO = 0;
        AUX_NUMERO_01 = Float.parseFloat(this.NUMERO_01.getText());
        AUX_NUMERO_02 = Float.parseFloat(this.NUMERO_02.getText());
        if (this.SUMAR.isSelected())
            {AUX_RESULTADO = AUX_NUMERO_01 + AUX_NUMERO_02;}
        else if (this.RESTAR.isSelected())
            {AUX_RESULTADO = AUX_NUMERO_01 - AUX_NUMERO_02;}
        else if (this.MULTIPLICAR.isSelected())
            {AUX_RESULTADO = AUX_NUMERO_01 * AUX_NUMERO_02;}
        else if (this.DIVIDIR.isSelected())
            {AUX_RESULTADO = AUX_NUMERO_01 / AUX_NUMERO_02;}
        this.RESULTADO.setText(String.valueOf(AUX_RESULTADO));
    }
    catch (Exception E)
    {
        this.RESULTADO.setText("ERROR AL EJECUTAR");
    }
}

public static void main(String[] ARGUMENTOS)
{
    Ej5 miGui = new Ej5();
    miGui.pack();
    miGui.setResizable(false);
    miGui.setVisible(true);
}
```

Ejercicio 6

```
public class Ejer6 extends JFrame implements ActionListener
{
    public JButton VERDE, AMARILLO, ROJO;
    public JPanel COLOR_PANEL;
    public Ejer6()
    {
        super("COLORES... Curso JAVA");
        this.setSize(400,400);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.VERDE = new JButton("VERDE");
        this.VERDE.setBackground(Color.GREEN);
        this.VERDE.addActionListener(this);
        this.AMARILLO = new JButton("AMARILLO");
        this.AMARILLO.setBackground(yellow);
        this.AMARILLO.addActionListener(this);
        this.ROJO = new JButton("ROJO");
        this.ROJO.setBackground(Color.RED);
        this.ROJO.addActionListener(this);
        this.COLOR_PANEL = new JPanel();
        this.COLOR_PANEL.setPreferredSize(new Dimension(300,300));
        this.COLOR_PANEL.setBackground(Color.BLACK);
        FlowLayout DISTRIBUIDOR_FRAME = new FlowLayout();
        this.setLayout(DISTRIBUIDOR_FRAME);
        this.add(this.VERDE);
        this.add(this.AMARILLO);
        this.add(this.ROJO);
        this.add(this.COLOR_PANEL);
    }
}
```

Ejercicio 6

```
public void actionPerformed(ActionEvent EVENTO)
{
    Object BOTON_SELECCIONADO = EVENTO.getSource();

    if(BOTON_SELECCIONADO == this.VERDE)
    {
        this.COLOR_PANEL.setBackground(Color.GREEN);
    }
    else if(BOTON_SELECCIONADO == this.AMARILLO)
    {
        this.COLOR_PANEL.setBackground(Color.YELLOW);
    }
    else if(BOTON_SELECCIONADO == this.ROJO)
    {
        this.COLOR_PANEL.setBackground(Color.RED);
    }
}

public static void main(String[] ARGUMENTOS)
{
    Ejer6 miGui = new Ejer6();
    miGui.setResizable(false);
    miGui.setVisible(true);
}
}
```

SWING

- *Cambiar el Aspecto del GUI: Swing tiene la posibilidad de definir un aspecto propio e independiente de la plataforma. En otras palabras, con Swing una misma GUI puede verse con distintos estilos. Para esto, Java provee las clases WindowLookAndFeel, MotifLookAndFeel y MetalLookAndFeel.*
- *Podemos cambiar el aspecto. Lo hacemos agregando una línea en el método main. Para ello hay que importar el paquete javax.swing.**

```
public static void main(String args[]) throws Exception
{
    UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
    ChatSwing c = new ChatSwing();
}
```

- *Para obtener los diferentes aspectos disponibles en nuestra plataforma se puede usar el siguiente código:*

```
Import javax.swing.UIManager.*;
LookAndFeelInfo[] lista = UIManager.getInstalledLookAndFeels();
for (int i = 0; i < lista.length; i++) {
    System.out.println(lista[i].getClassName());
}
```


Ejercicio

- Hacer un programa, que contenga un menú para poder elegir entre los ejercicios hechos en le curso
 - ▣ Recursividad
 - ▣ Segundo pack de ejercicios
 - ▣ Ejercicios Gráficos
- Después de elegir la opción muestre los ejercicios disponibles con un `RadioButton` donde se seleccione cual ejecutar y un botón que lo ejecute.
- Utilizar los `JDialog` para pedir los datos y mostrar el resultado

SWING

- Hacer una calculadora:

