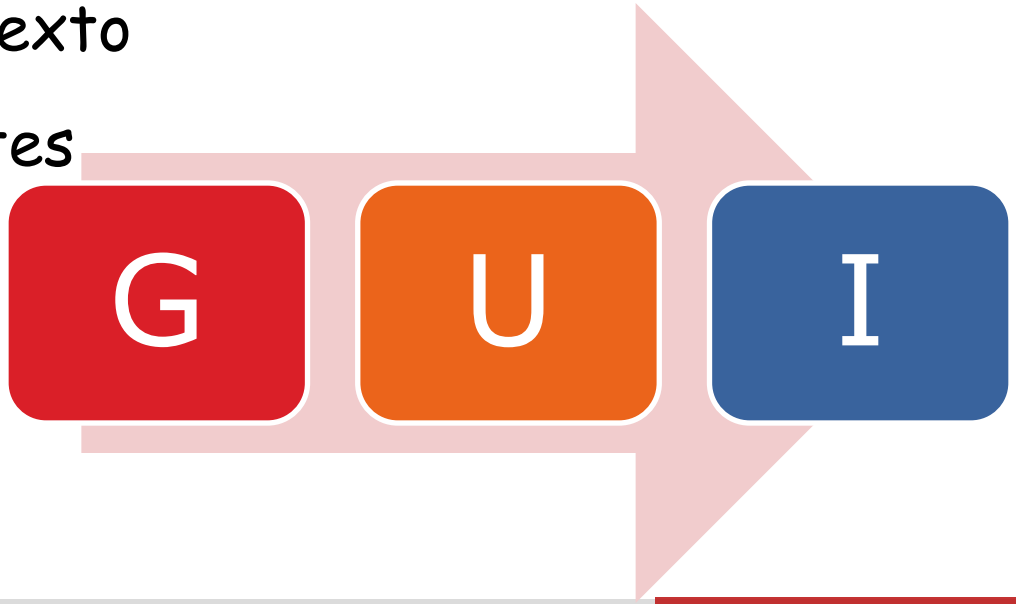
INTERFACES GRÁFICAS DE USUARIO EN JAVA (GUI)

Componentes Swing

Universidad del Valle

CONTENIDO

- Definiciones
- Componentes de Swing
  - Contenedores
  - Componente atómicos
  - Componente de texto
  - Otros componentes
- Ejemplos
- Ejercicios



GUI – DEFINICIONES

Programación Orientada a Eventos

- Los programas con GUI son "event-driven"
  - La secuencia de las entradas controlan el flujo de la ejecución
  - Los eventos producidos en los componentes invocan código del usuario.

GUI – DEFINICIONES

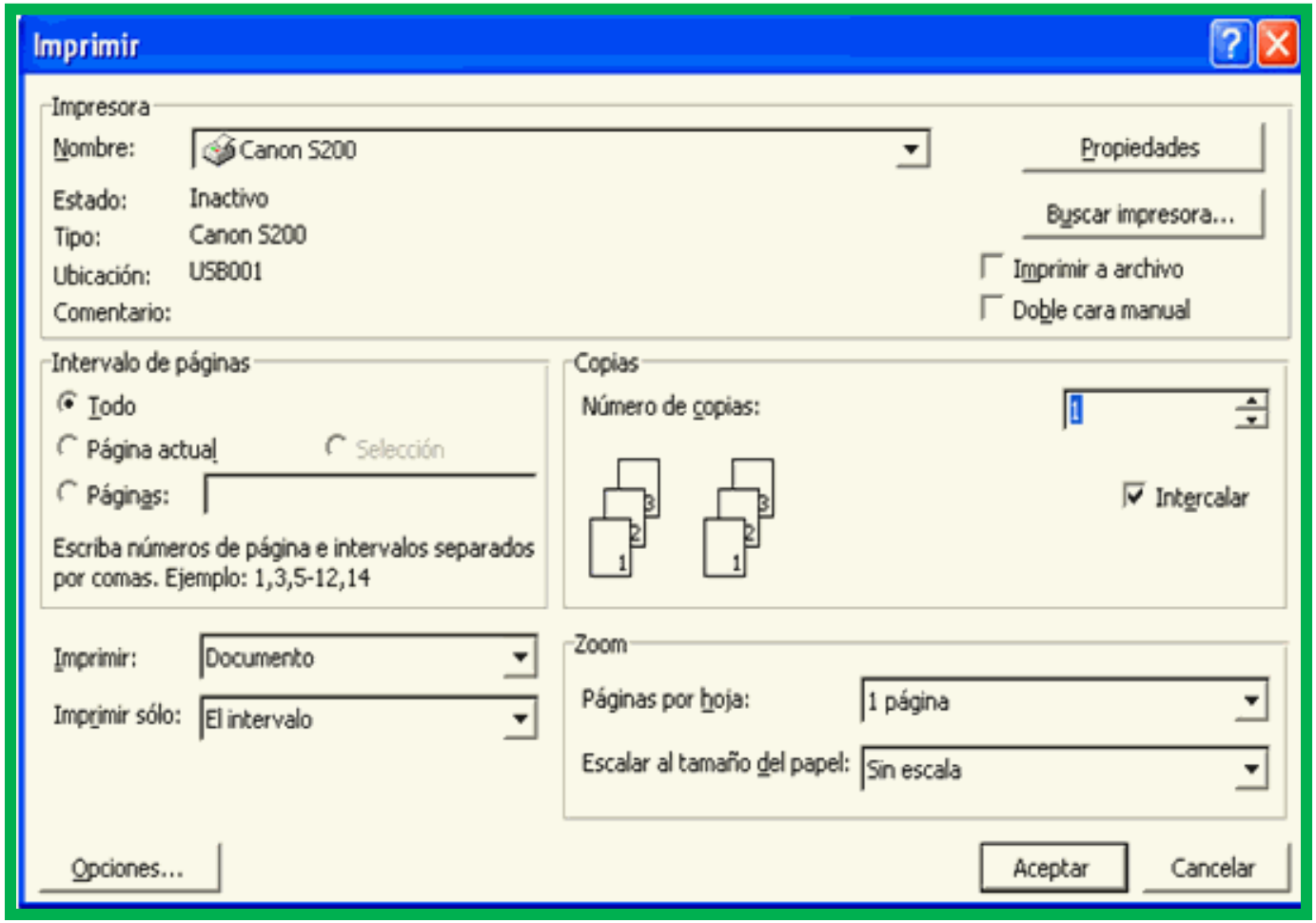
Interfaz gráfica de usuario

Aplicaciones que presentan una interfaz gráfica avanzada con el usuario, es decir, aquellas aplicaciones que incluyen:

Las GUI permiten presentar un modo amigable al usuario para interactuar con un programa

GUI – Interfaces Gráficas de Usuario

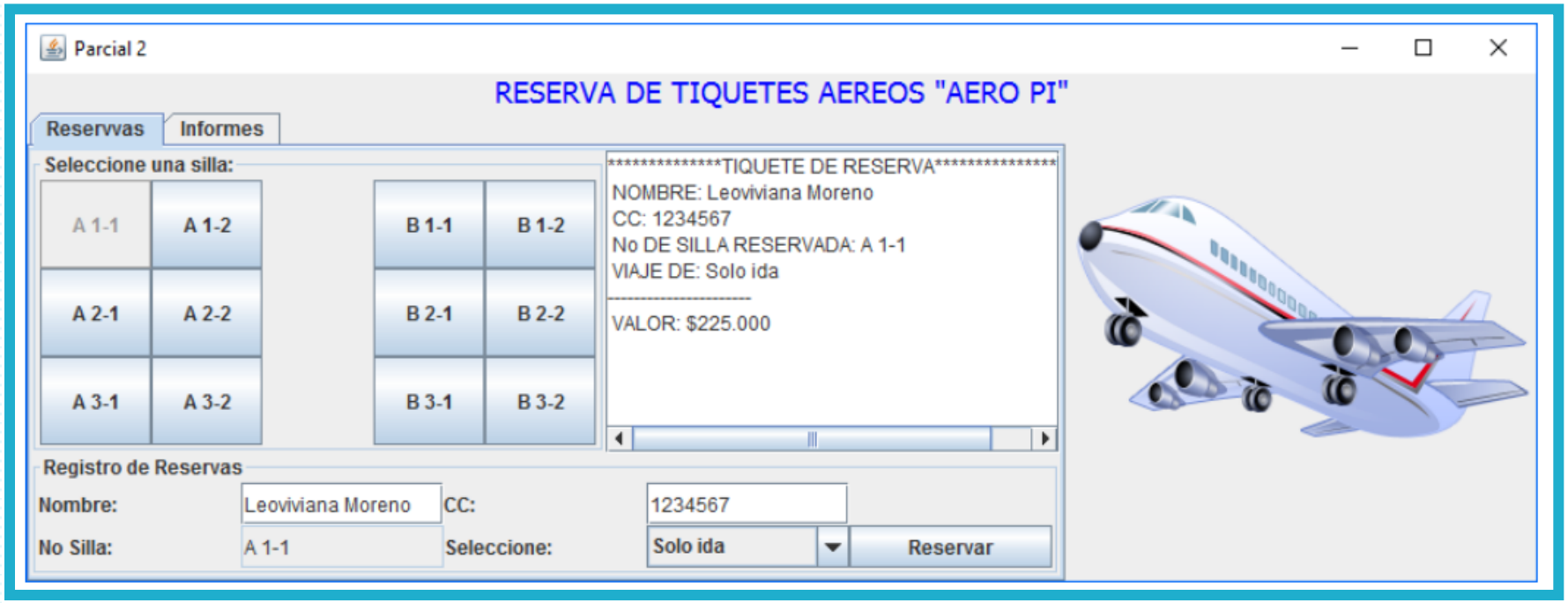
GUI – Interfaces Gráficas de Usuario



GUI – Interfaces Gráficas de Usuario

GUI – Interfaces Gráficas de Usuario

GUI – Interfaces Gráficas de Usuario



GUI – LIBRERÍAS

Java provee librerías para crear GUIs, a ese conjunto de librerías se le conoce como API's. De las más usadas están:

- ✓ Java Swing
- ✓ Java 2D
- ✓ Java Accessibility
- ✓ Drag and Drop

Ver ejemplos en Campusvirtual

GUI – LIBRERÍAS

- Componentes **Swing**: implementan un conjunto de componentes de GUI con look & feel configurable.
- **Java 2D**: es un conjunto de clases que permiten manipular gráficos 2D avanzados e imágenes.
- **Accesibility**: consiste en un conjunto de clases que permiten a las componentes Swing interactuar con tecnologías que ayudan a los usuarios con discapacidades como lectores de pantalla y display Braille.
- **Drag and Drop**: Proporciona la habilidad de arrastrar y soltar componentes en aplicaciones Java.

GUI –Aplicaciones basadas en GUI

El desarrollo de una aplicación basada en GUI requiere la comprensión de:

- **Estructura de la jerarquía de herencia**, que define el comportamiento y los atributos de los componentes en la GUI en la aplicación.
- **Estructura de la jerarquía de contenedores y gestores de distribución**, que definen cómo se disponen todos los componentes en la GUI de la aplicación.
- **Manejo de eventos**.

Componentes de Swing

- La clase **Component** (y sus subclases) proveen soporte para manejo de eventos, cambio del tamaño de un componente, control de color y fuentes.

Un componente es un objeto de una clase concreta.

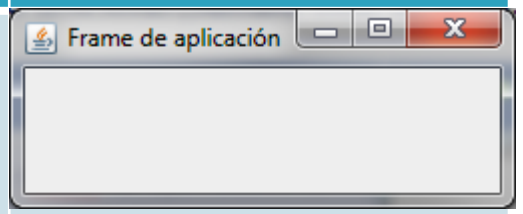
Se distinguen dos clases de componentes:

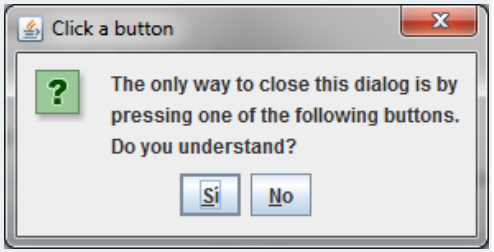
- **Componentes de control** de la GUI: la interacción de la GUI con el usuario se realiza a través de ellos.
- **Contenedores**: contienen otros componentes (u otros contenedores).

CONTENEDORES

- Anidamiento de componentes (*Jerarquía de contenedores*). Cada programa Swing contiene al menos uno.
- Swing provee contenedores de alto nivel **(ventana base de la GUI):**

Contenedor	Creación	Imagen

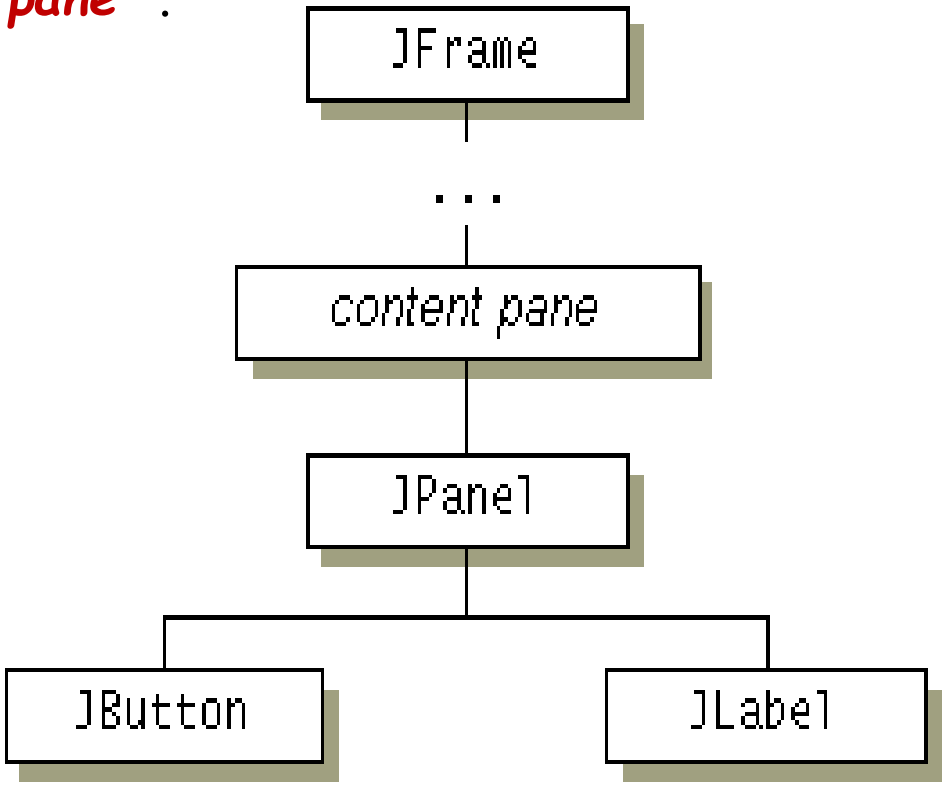
JFrame	`JFrame fr = new JFrame("Frame de aplicación");`
	`fr.setSize(300, 300);  //tamaño del frame`
	`fr.setVisible(true); //mostrar el frame`
		

JDialog	ventanas más limitadas que los frames, Para crear un diálogo, se utiliza **JOptionPane** ó JDialog **JOptionPane.showXXXDialog**(…..);	

JWindow	`JWindow w = new JWindow();`
	`w.setSize(300, 300);`
	
	`w.setVisible(true);`	

CONTENEDORES

- La jerarquía está compuesta de diferentes capas.
- Cada contenedor de alto nivel contiene **un contenedor intermedio** (capa) conocido como "***content pane*** ".



CONTENT PANE

Son contenedores intermedios que tienen como funcionalidad **contener a otros componentes de la interfaz**. Algunos de los cuales son:

componente	Definición	Creación
Container	Contenedor de AWT, de quien heredan todos los contenedores de alto nivel de Swing. Es el contentPane por defecto.	Dentro del constructor de la clase que hereda de JFrame se puede utilizar de la siguiente manera: **Container cont = getContentPane();**

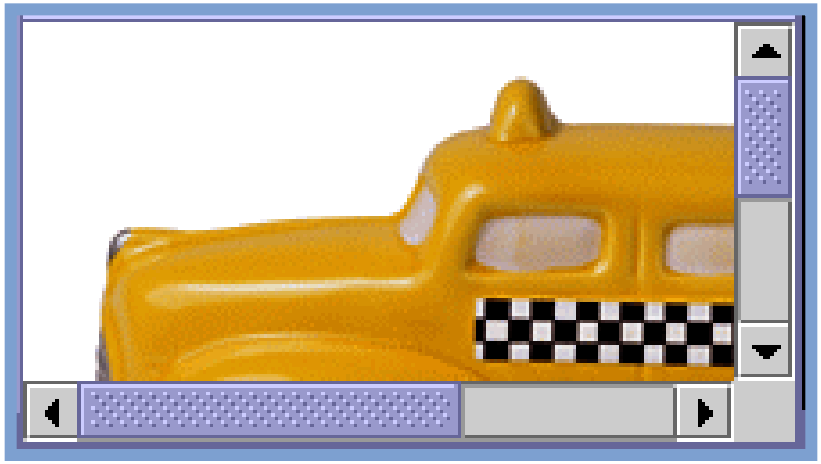
Panel	son contenedores de propósito general que sirve para agrupar componentes en la interfaz	**Jpanel** panel = **new Jpanel**();
		panel.**setSize**(200,200); //para dar tamaño
		//se adiciona al contenedor principal
		contenedorPpal.**add(panel);**

Scroll pane	proporciona barras de desplazamiento alrededor de un sólo componente	**JScrollPane** sc= **new JScrollPane**(componente);

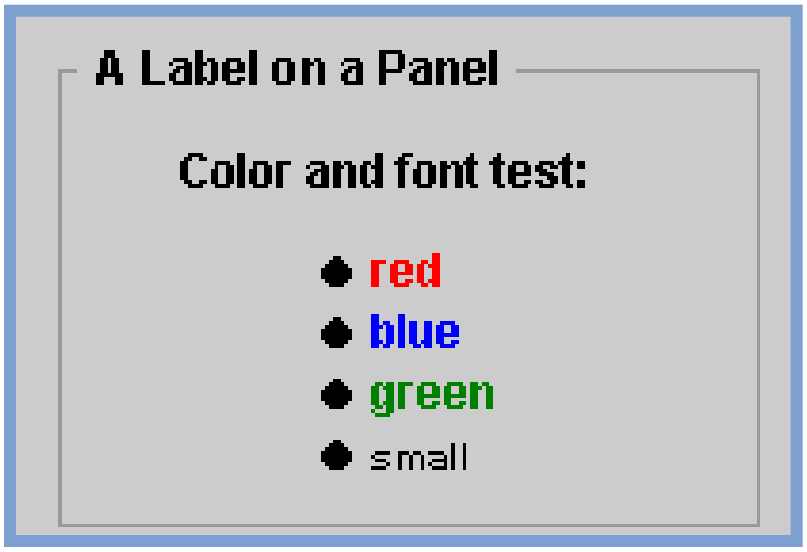
Split Pane	permite al usuario personalizar la cantidad relativa de espacio dedicada a cada uno de dos componentes	// crear split con 2 scrollpane dentro
		SplitPane sp = new SplitPane(JsplitPane.HORIZONTAL_SPLIT);
		splitPane.setLeftComponent(scroll1);
		splitPane.setLEFTComponent(scroll2);
		splitPane.setOneTouchExpandable(true);

CONTENEDORES INTERMEDIOS

Scroll Pane



Panel



SplitPane

CONTENT PANES

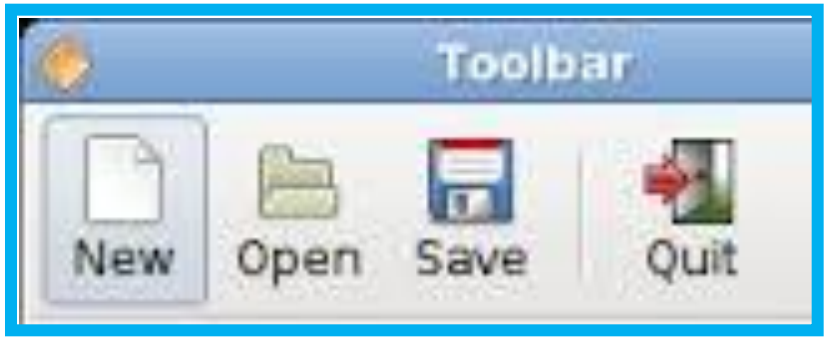
componente	Definición	Creación

Tool bar	contiene un grupo de componentes (normalmente botones) en una fila o columna	//toolbar con 2 botones ubicado al norte contenedor
		JToolBar toolBar = **new JToolBar();**
		toolBar.**add(new JButton("1"));**
		toolBar.**add(new JButton("2"));**
		contenedor.**add(toolBar, BorderLayout.NORTH**);

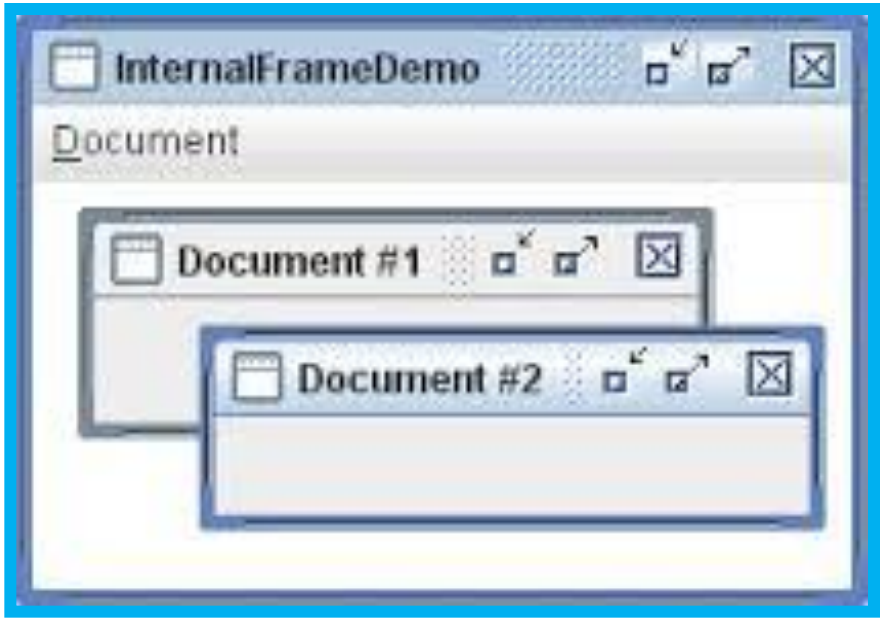
Internal frames	Similares a los Frames, pero deben aparecer dentro de otras ventanas	**JInternalFrame** internal = **new JInternalFrame("IFrame");** **internal.add(new Jpanel());** // darle tamaño -pack()- al JInternalFrame, para verlo. **internal.pack();** contenedor.**add(internal);** **internal.setVisible(true);**
Tabbed pane	Pestañas dentro de una misma ventana	**//tablet pane con un panel dentro** **JTabbedPane** tp = new **JTabbedPane()**; **tp.add(new Jpanel());** contenedorPpal.**add(tp**);

CONTENEDORES INTERMEDIOS

TabbedPane



Toolbar



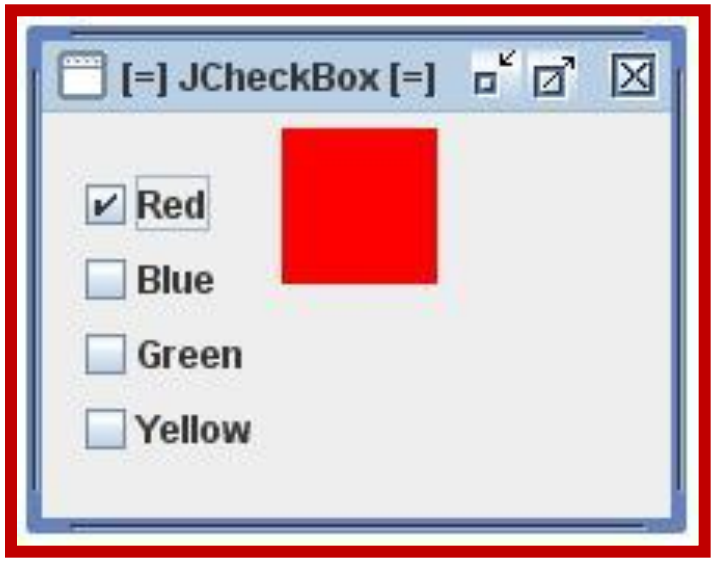
Internal Frame

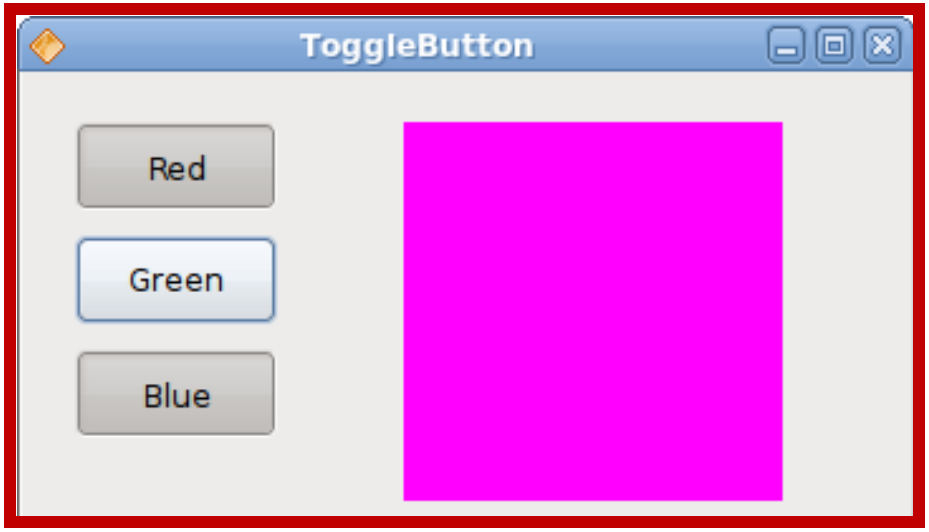
COMPONENTES ATÓMICOS

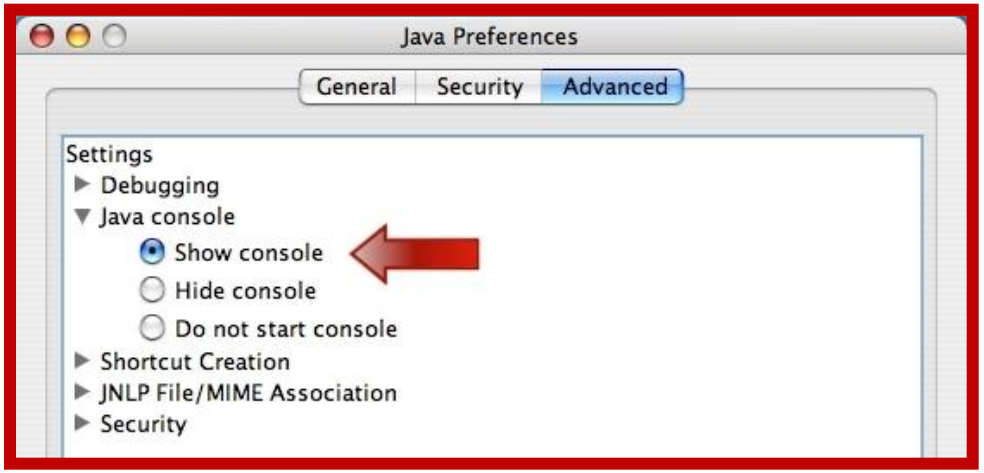
Los componentes atómicos son los elementos que no pueden almacenar otros objetos o componentes gráficos. Algunos son:

Componente	Definición
JLabel	Permite crear etiquetas, tanto de texto como de imágenes
JButton	Permite crear botones simples
JCheckBox	Son casilla de verificación, para selección múltiples
JRadioButton	Permite presentar opciones de selección única.
JToggleButton	Botón que al oprimirlo se quedará presionado hasta que se ejecute otro evento.
JComboBox	Permite mostrar una lista de elementos como un combo de selección.

COMPONENTES ATÓMICOS



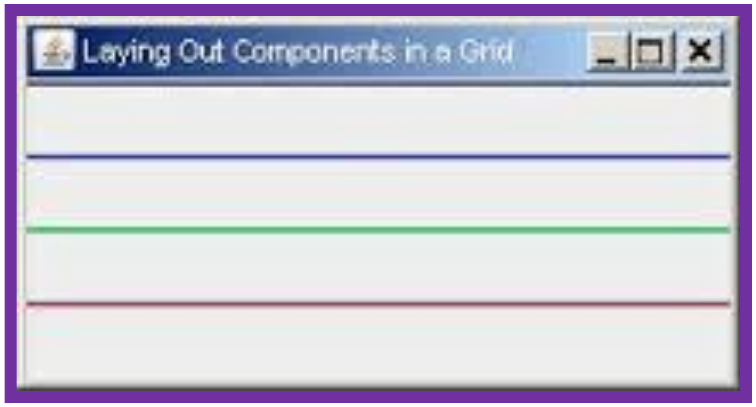


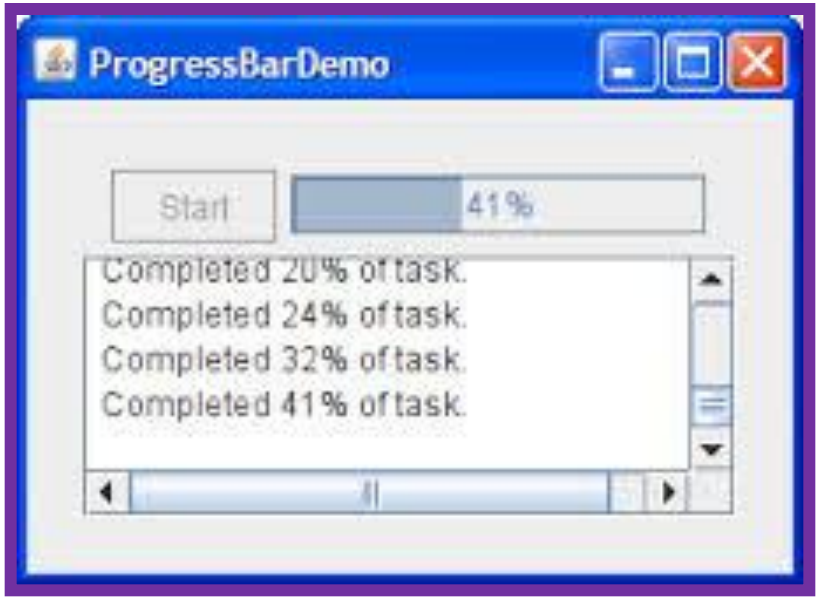


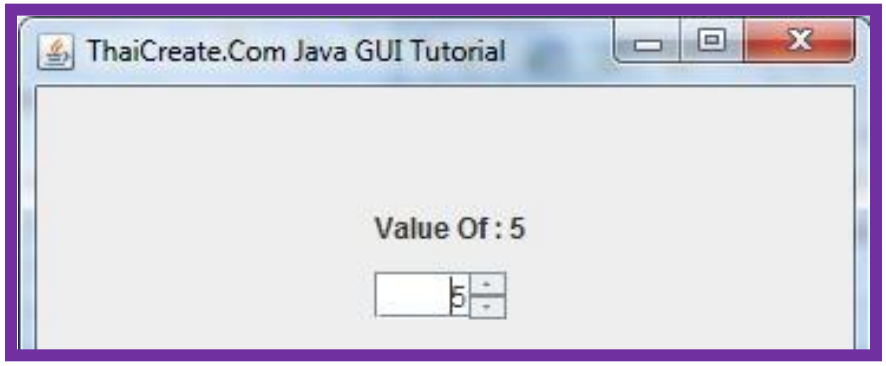
COMPONENTES ATÓMICOS

Componente	Definición
JSeparator	Permite separar opciones, es una línea divisoria simple
JSlider	Permite vincular un Deslizador en nuestra ventana
JSpinner	permite vincular una caja de texto con botones integrados para seleccionar algún valor
JProgressBar	Establece una barra de progreso.

COMPONENTES ATÓMICOS





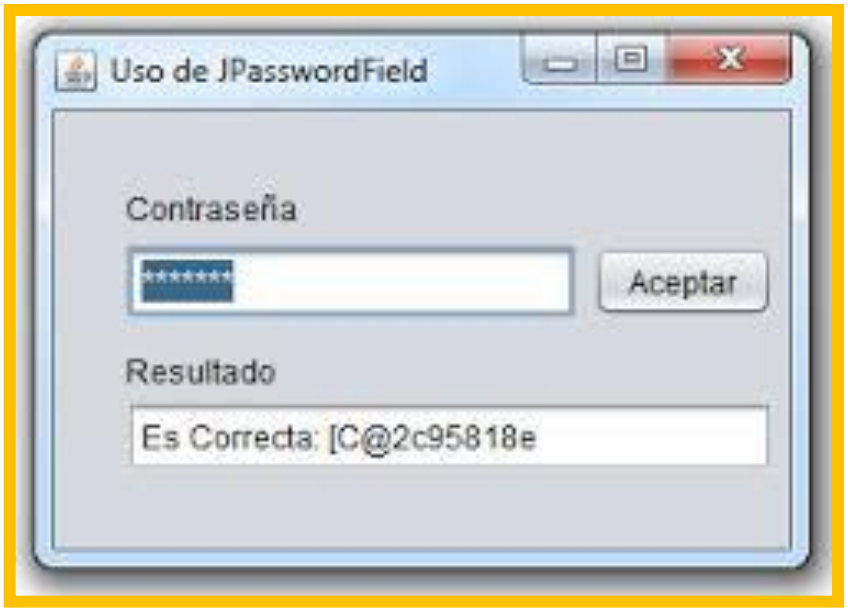


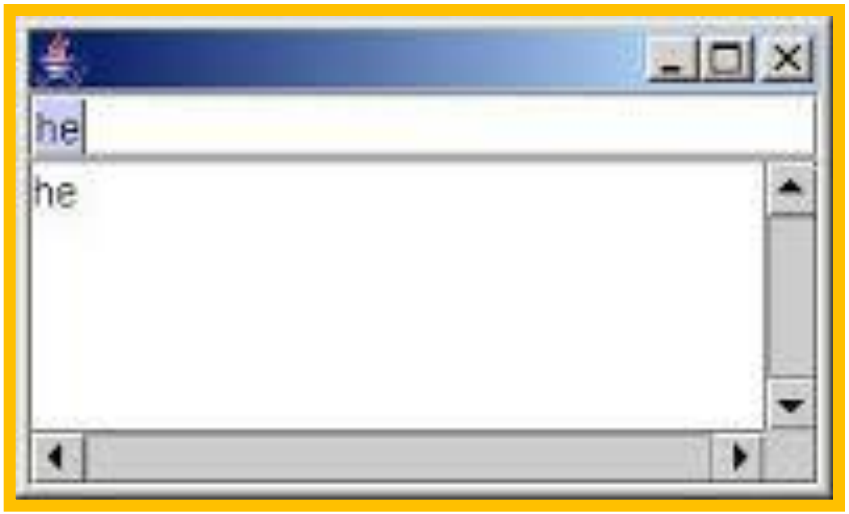
Componentes de Texto

Son todos aquellos que permiten procesar cadenas de texto, sea como entrada o salida de información. Algunos son:

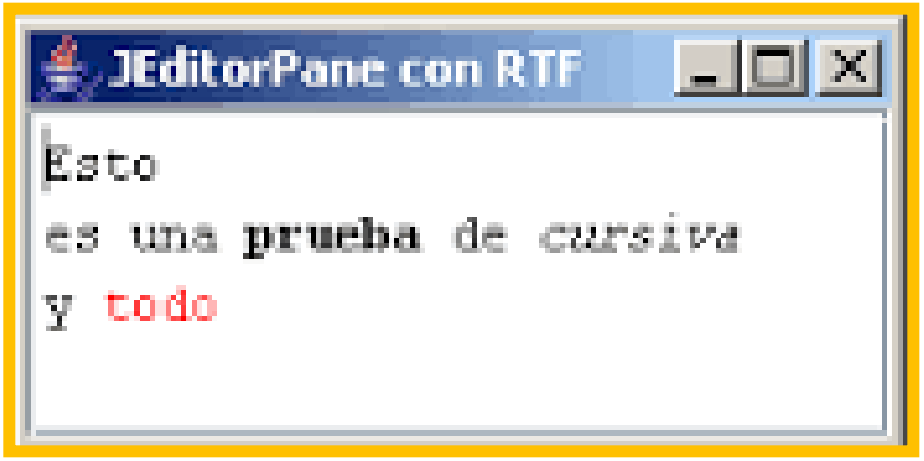
Componente	Definición
JTextField	Permite introducir un campo de texto simple
JFormattedTextField	Permite introducir un campo de texto con formato, (si definimos que solo recibe números no permitirá letras, etc.)
JPasswordField	Campo de texto que oculta los caracteres ingresados
JTextArea	Permite crear un área de texto donde el usuario puede ingresar o imprimir cadenas de texto
JEditorPane	Permite vincular un área de texto con propiedades de formato
JTextPane	Similar al anterior, permitiendo otras opciones de formato, colores, iconos entre otros.

Componentes de Texto





Componentes de Texto

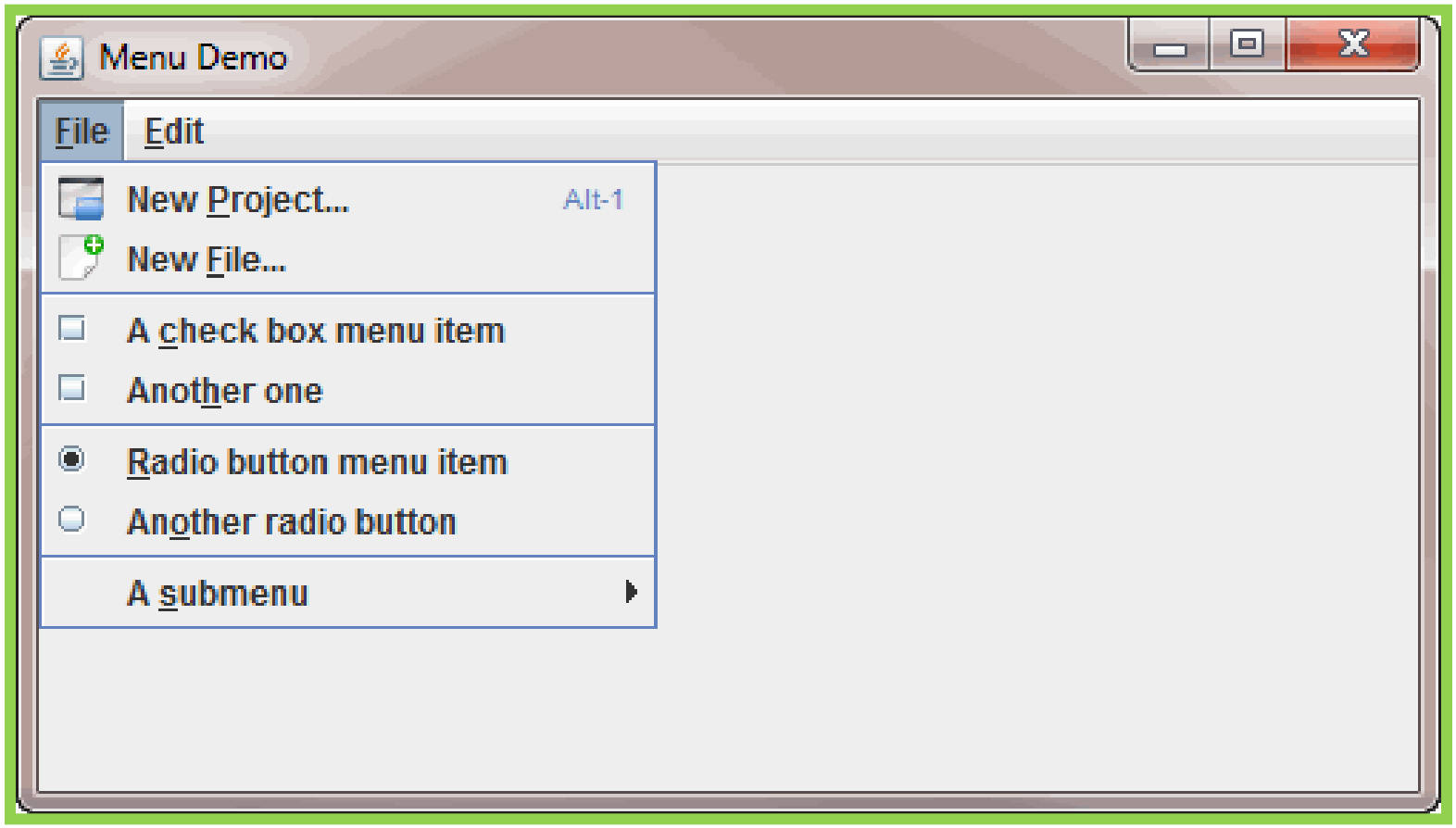


Componentes de Menú

Estos componentes permiten vincular opciones de menú en las ventanas, tipo menú principal, como por ejemplo el conocido Inicio, Archivo, Edición etc.

Componente	Definición
JMenuBar	Permite vincular una barra de menús.
JMenu	Permite vincular botones o enlaces que al ser pulsados despliegan un menú principal.
JMenuItem	Botón u opción que se encuentra en un menú.
JCheckBoxMenuItem	Elemento del menú como opciones de checkbox
JRadioButtonMenuItem	Elemento del menú como botón de selección
JPopupMenu	Opciones de menú emergentes.

COMPONENTE JMENU, JMENUITEM Y JMENUBAR

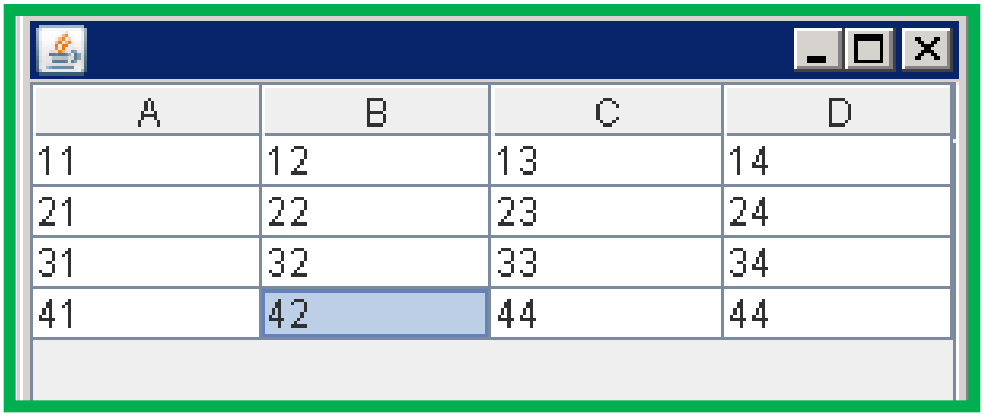


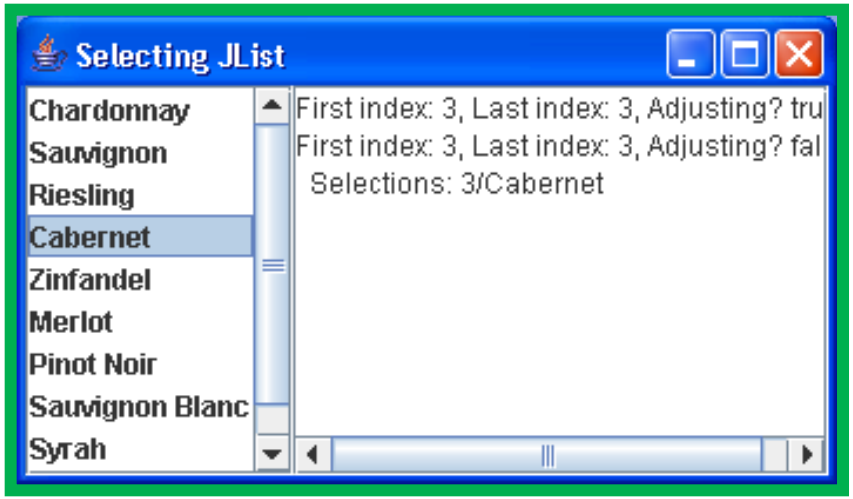
Componentes Complejos

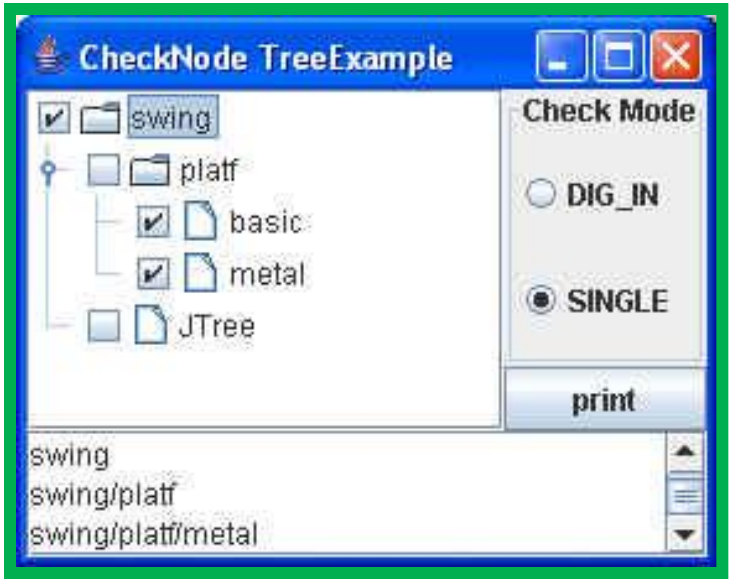
Estos son componentes un poco mas avanzados, cumplen con funciones mas enfocadas a procesos específicos y complejos, como por ejemplo obtener gran cantidad de información de una base de datos, trabajo con nodos, colores entre otros.

Componente	Definición
JTable	Permite vincular una tabla de datos con filas y columnas.
JTree	Carga un árbol donde se establece cierta jerarquía visual tipo directorio
JList	Permite cargar una lista de elementos.
JFileChooser	Componente que permite la búsqueda y selección de archivos
JColorChooser	Componente que permite cargar un panel selector de color

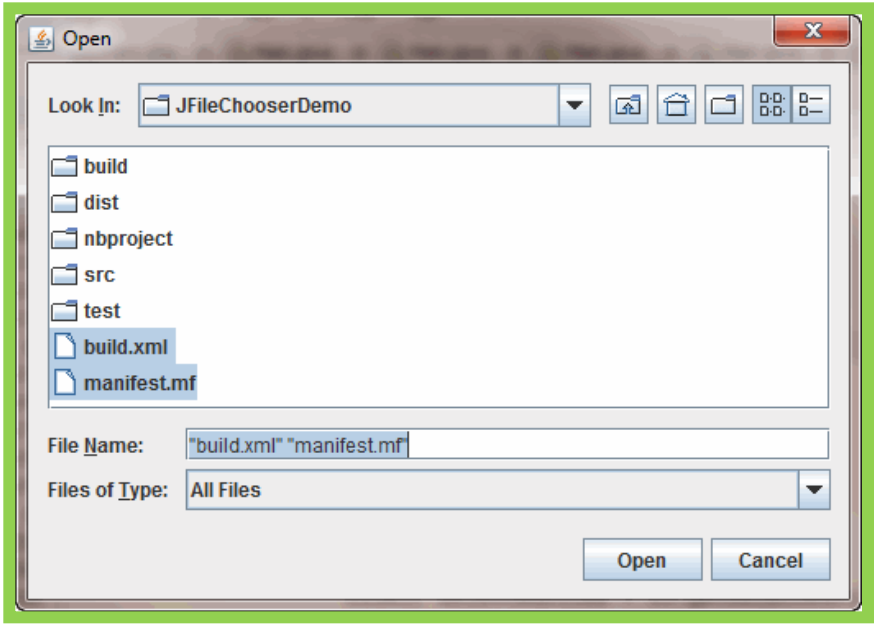
Componentes Complejos

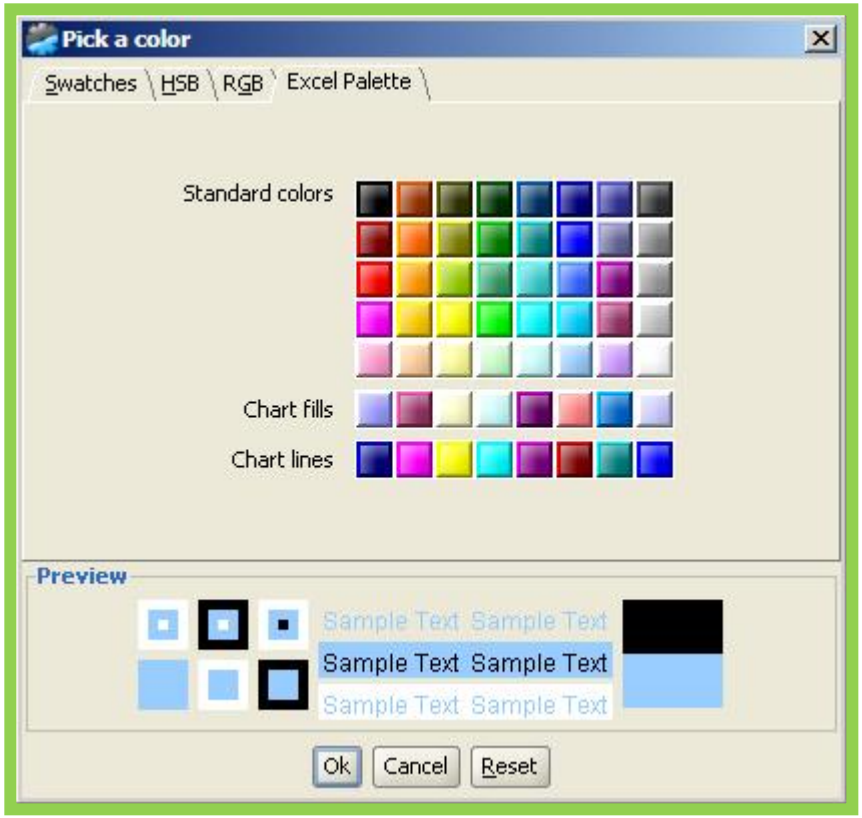






Componentes Complejos





APARIENCIA DE GUI DENTRO DE CONTENEDORES

- La apariencia de una GUI está determinada por:
  - La jerarquía de contenedores
  - El Layout Manager **(gestor de diseño o distribucción)** de cada contenedor
  - Las propiedades de los componentes individuales

Todos estos ítems trabajan en conjunto para determinar el efecto visual final.

Gestores de diseño (Layouts)

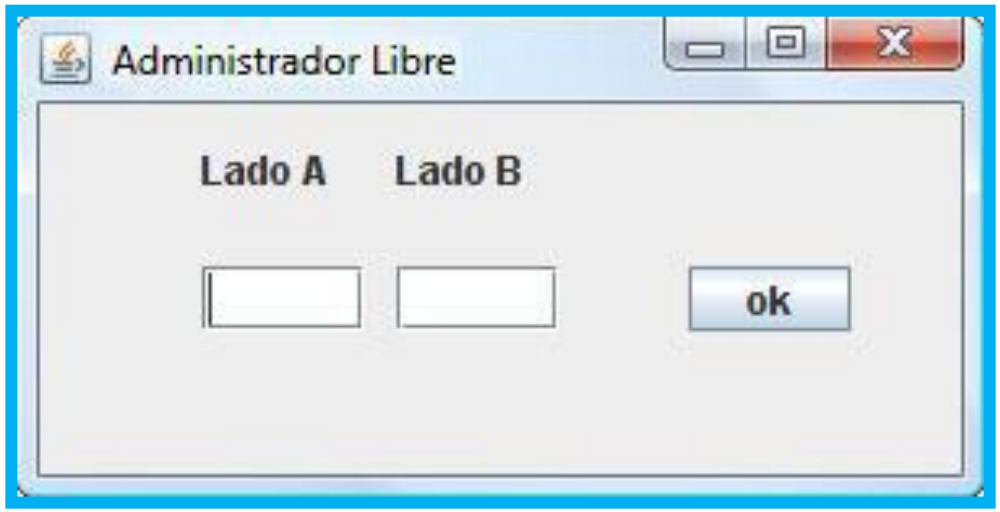
Los **Layout** son clases en Java, que se encargan de decidir, cómo van colocados los botones, áreas de texto y demás componentes dentro de un frame, panel, ventana o cualquier otro contenedor.

El Layout decide si los botones se pegan a la izquierda, a la derecha, si se deben hacer o no grandes al estirar la ventana, etc.

Gestores de diseño Libre

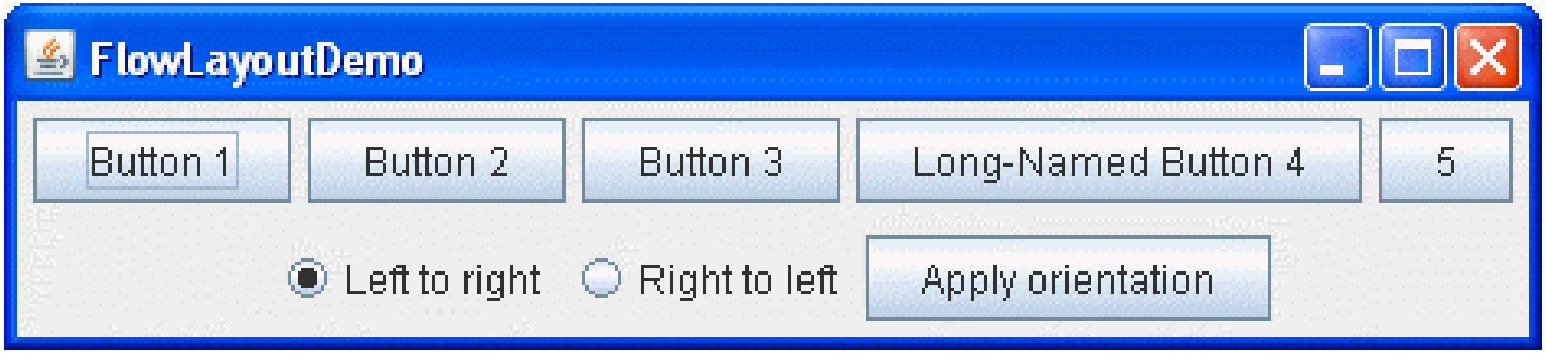
Se caracteriza en que entrega toda la responsabilidad para presentar los componentes en el frame al programador. Por defecto el administrador de diseño es FlowLayout.

En el Gestor de diseño Libre se insertan los componentes y se debe indicar la posición física de cada uno dentro del Frame. Estos **son los que usan por defecto los IDE de generación automática de interfaces**



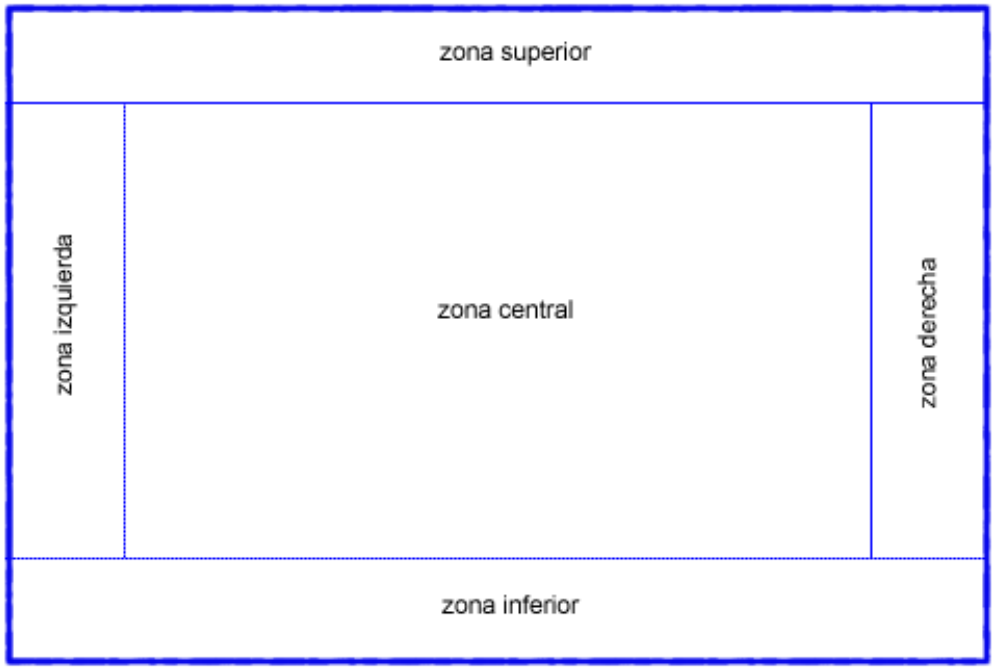
GESTORES DE DISEÑO FLOWLAYOUT

FlowLayout es el controlador por defecto para todos los Paneles. Simplemente coloca los componentes de izquierda a derecha, empezando una nueva línea si es necesario

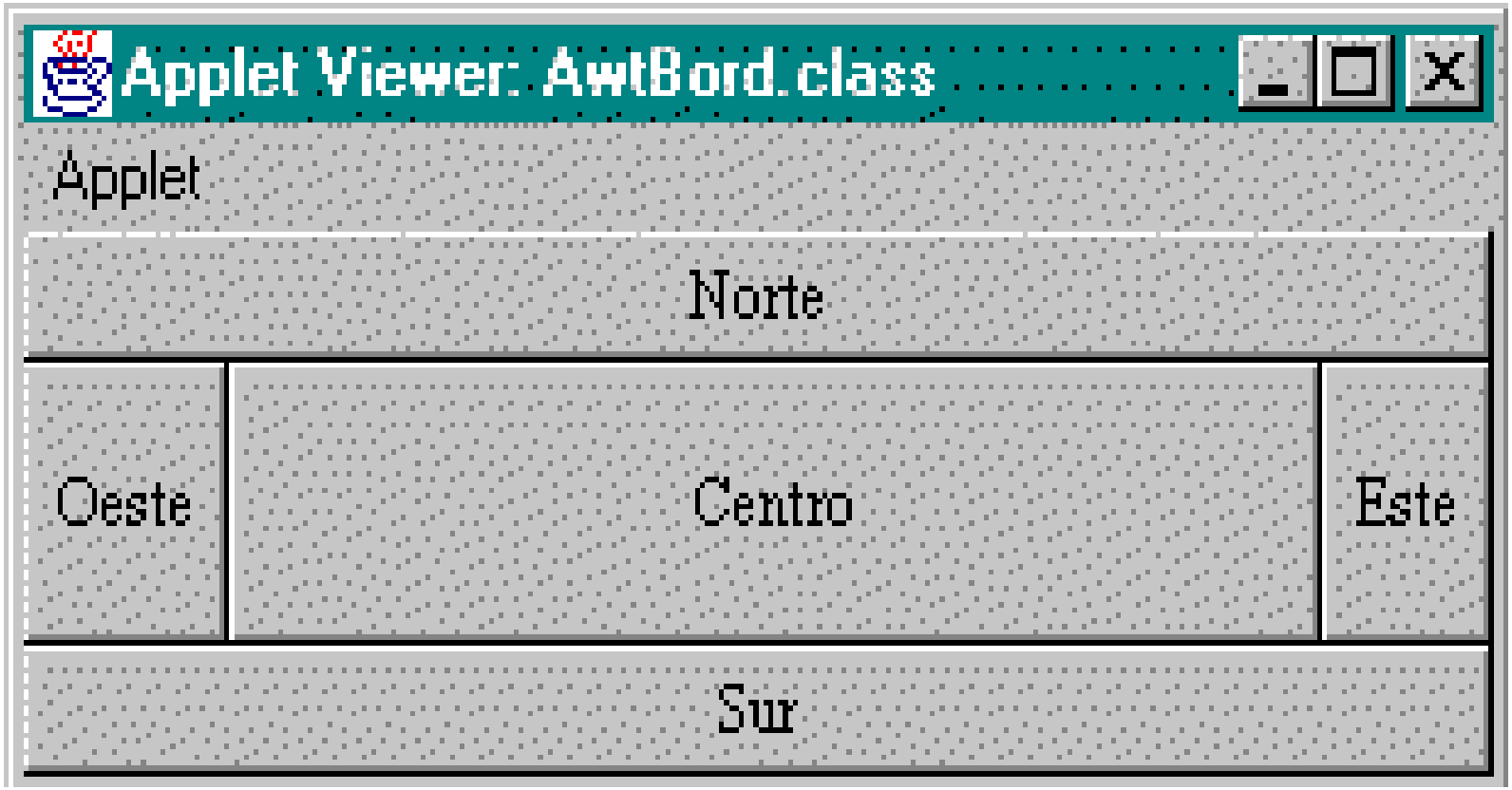


GESTORES DE DISEÑO BORDERLAYOUT

El **BorderLayout**. es un **Layout** que divide el panel en cinco zonas: una central, una arriba, otra abajo, en la izquierda y en la derecha. El tamaño de estas zonas depende de los componentes que van dentro

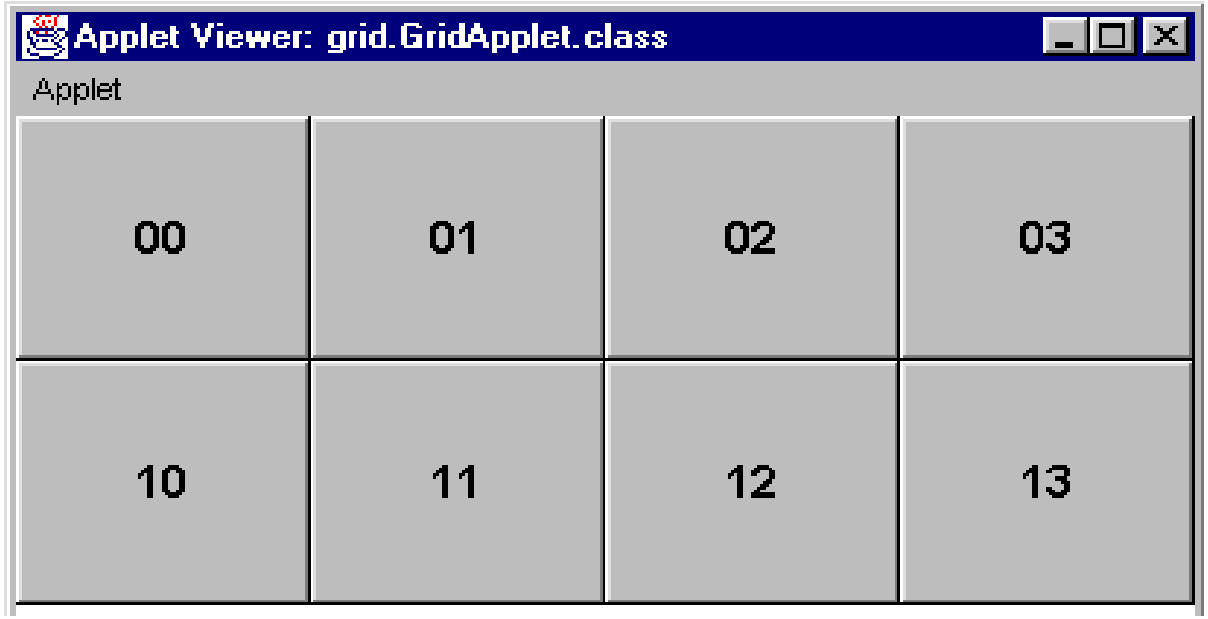


GESTORES DE DISEÑO BORDERLAYOUT



GESTORES DE DISEÑO GRIDLAYOUT

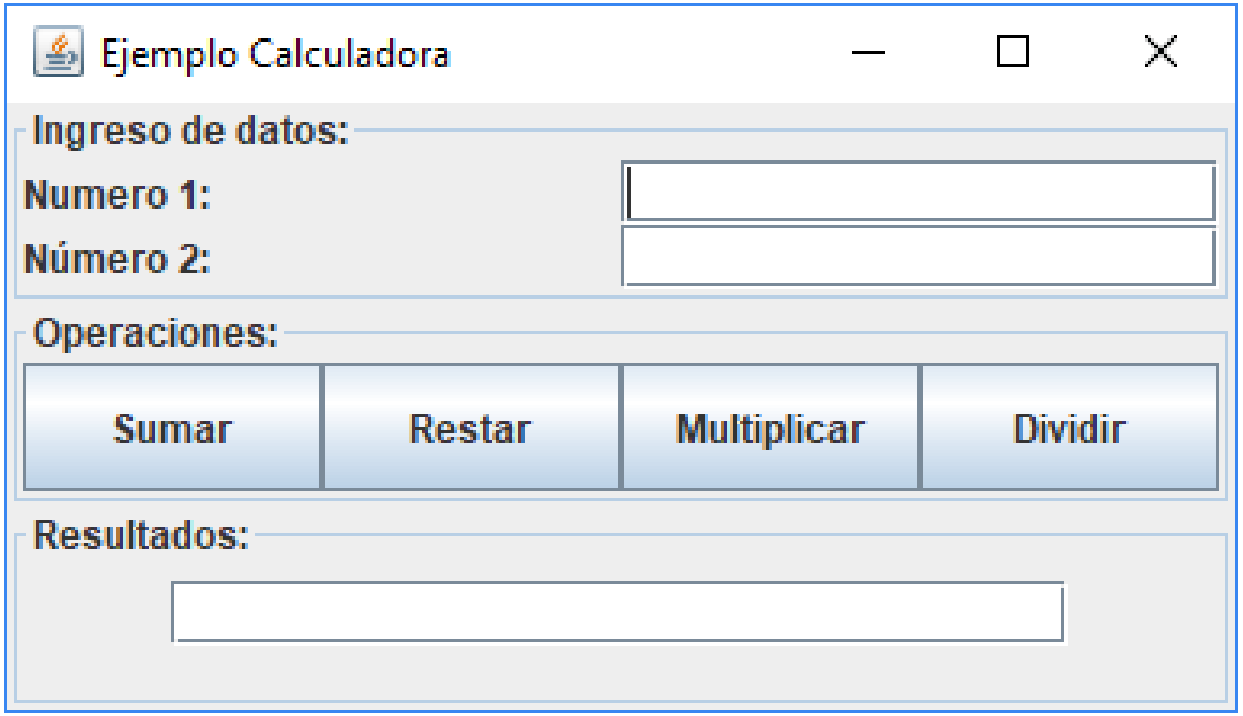
El **GridLayout**. es un **Layout** que divide el panel en forma de una tabla ó grilla , en el momento de la creación se especifica cual la cantidad de filas y de columnas a crear.



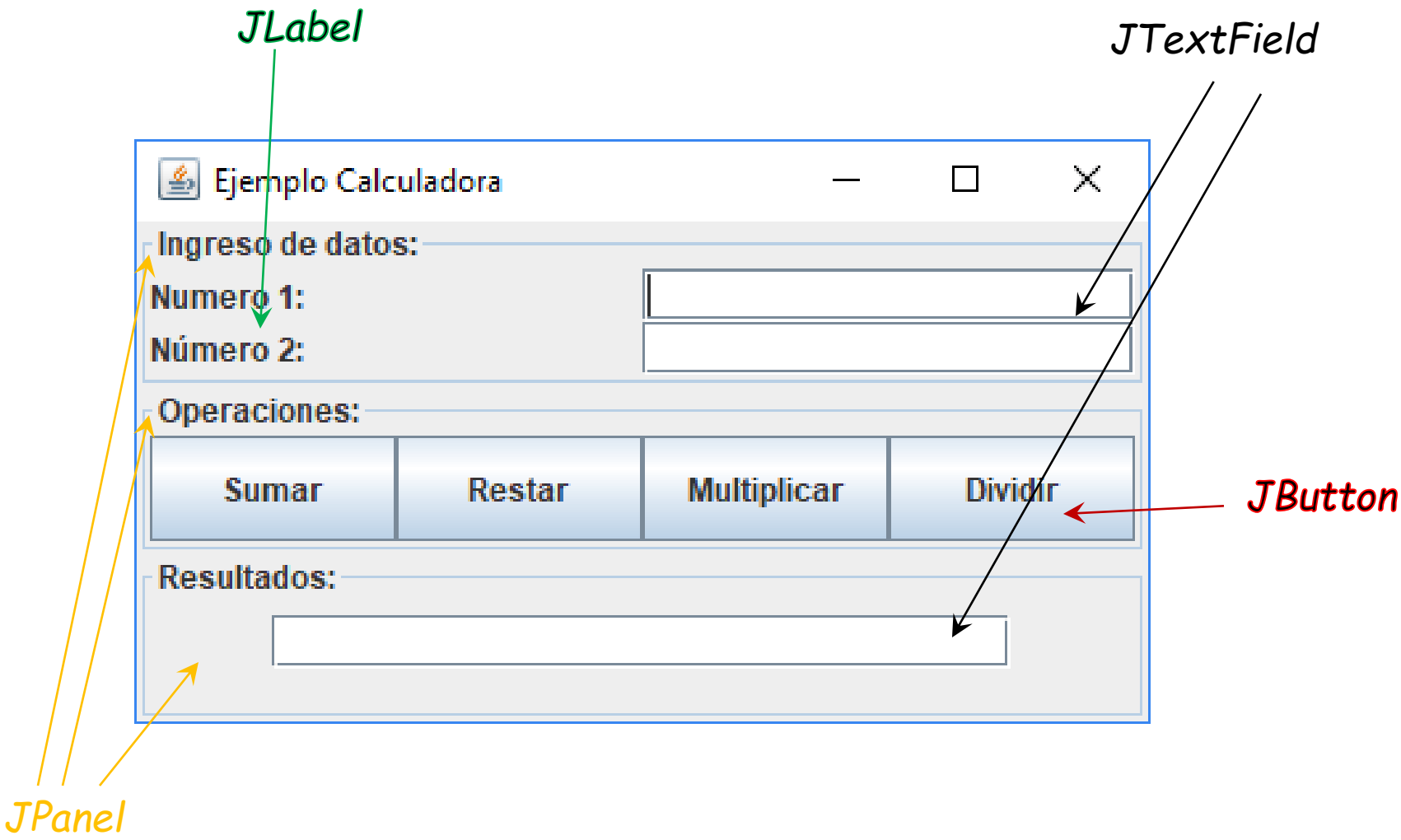
Este GridLayout tiene ocho botones dispuestos en dos filas y en cuatro columnas.

EJEMPLOS:

Interfaz Sencilla

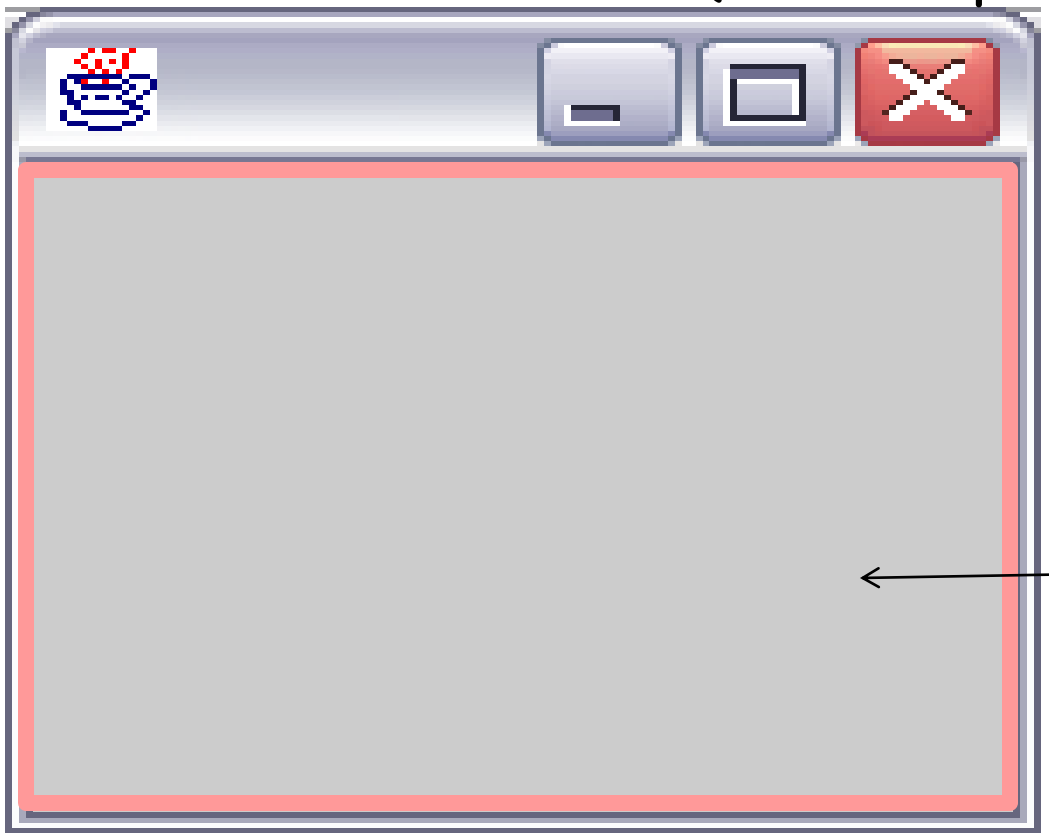


Interfaz Sencilla

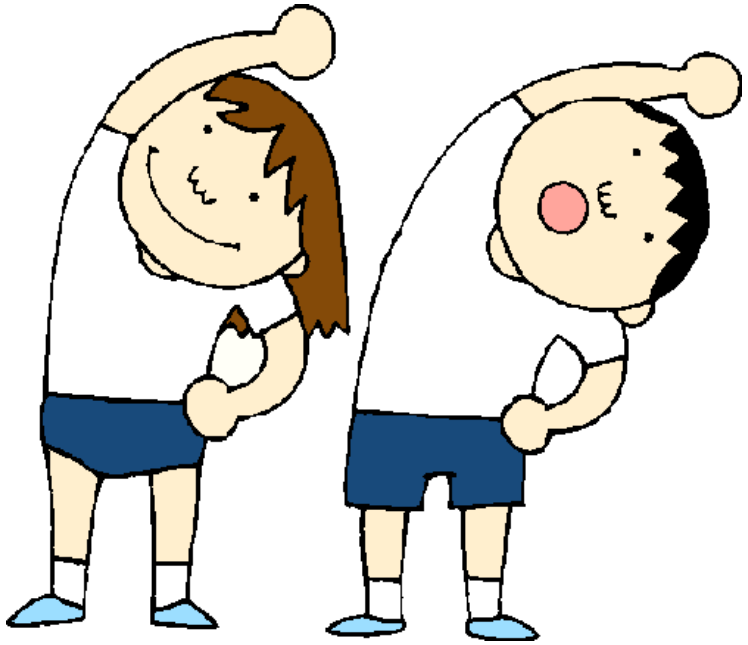


Interfaz Sencilla

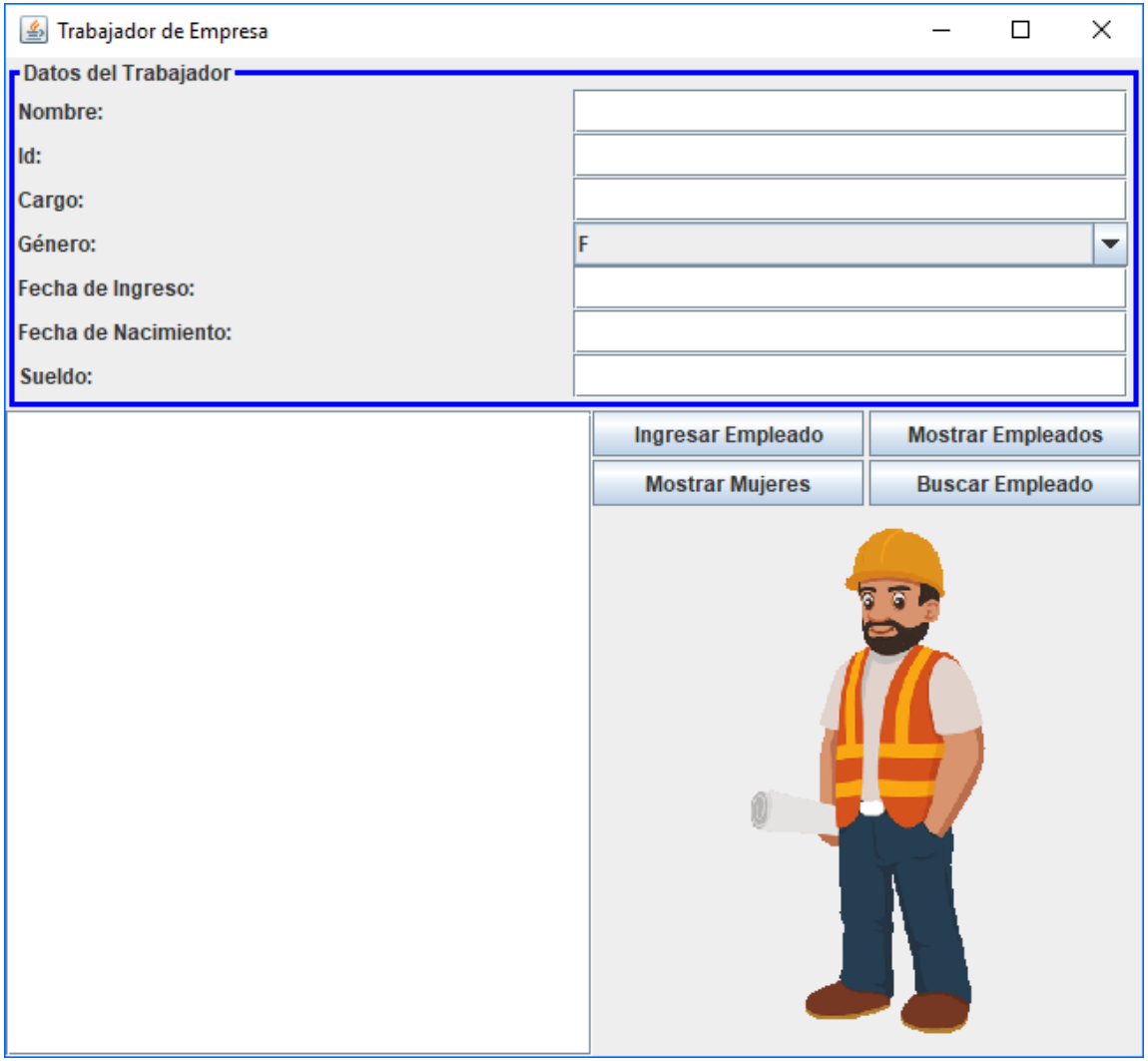
Cada elemento que hará parte del JFrame, se debe adicionar a él (content pane).



EJERCICIOS…



DESARROLLAR LAS SIGUIENTE GUI



DESARROLLAR LAS SIGUIENTE GUI

DESARROLLAR LAS SIGUIENTE GUI

ALGUNAS COSITAS..

```
Jlabel etiqueta = new JLabel("Etiqueta e Icono", new
ImageIcon(getClass.getResource("ship_032.gif")), SwingConstants.RIGHT);

getContentPane().add(etiqueta, BorderLayout.CENTER);
```

Botones JButton	
Creación	*JButton nombreB = new Jbutton(«mensaje»);*
Mostrar texto	*nombreB .setText(«texto a mostrar»);*
Tomar valor	*nombreB .getText();*
Poner Ícono	nombreB.setIcon(new ImageIcon(«nombreImagen.extension"));

ALGUNAS COSITAS..

Un componente Jpanel, es un contendor de componentes; se pueden incluir otros componentes dentro del panel.

JPanel()
Crea un contenedor de componentes (panel), con el administrador de diseño FlowLayout.

JPanel(LayoutManager admon)
Crea un panel, con el administrador de diseño indicado.


```
JButton ok = new JButton("Calcular");
JPanel panel = new JPanel();
panel.add(ok);
panel.setBorder(BorderFactory.createTitledBorder("Panel"));
getContentPane().add(panel);
```

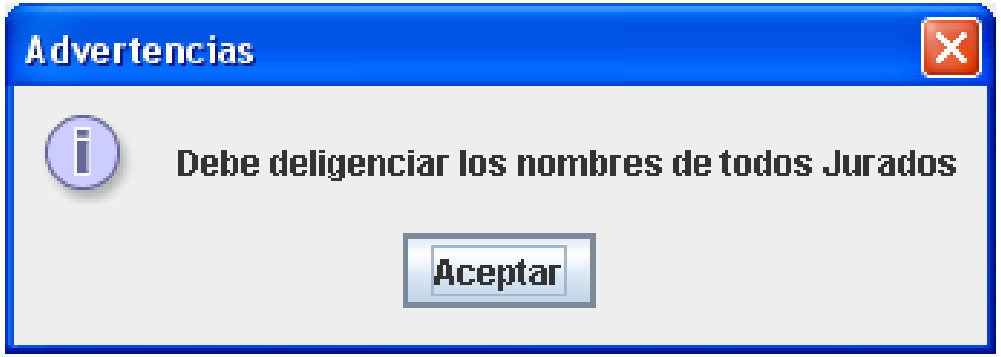
Con color y grosor de la línea del borde:

```
panel.setBorder(BorderFactory.createTitledBorder(BorderFactory.create
LineBorder(Color.BLUE, 3),"Titulo del borde del panel"));
```

ALGUNAS COSITAS..

JOPtionPane

Mensaje de advertencia:



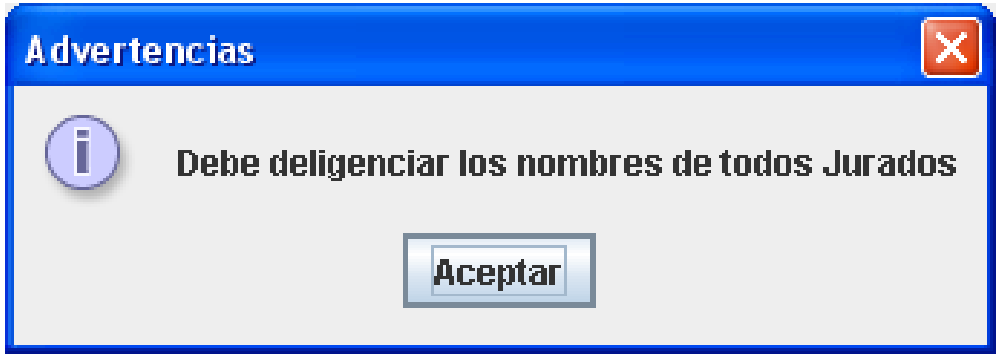
Mensaje que se muestra en el diálogo

```
JOptionPane.showMessageDialog(this,
                "Debe deligenciar los nombres de todos Jurados",
                "Advertencias",
                JOptionPane.INFORMATION_MESSAGE);}
```

ALGUNAS COSITAS..

JOPtionPane

Mensaje de advertencia:



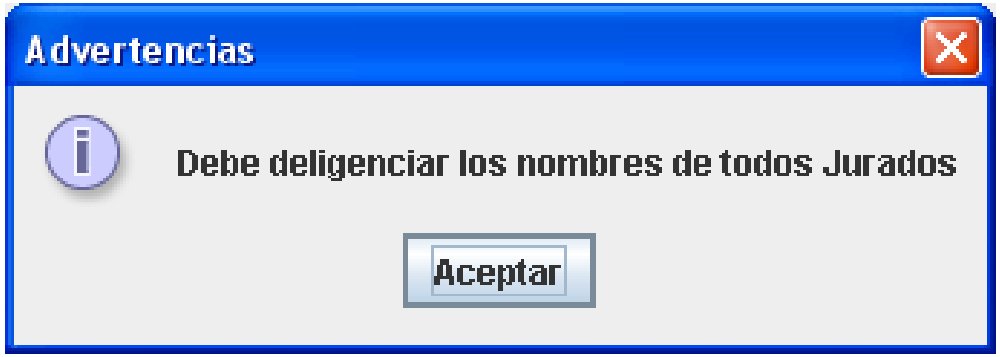

```
JOptionPane.showMessageDialog(this,
                    "Debe deligenciar los nombres de todos Jurados",
                    "Advertencias",
                    JOptionPane.INFORMATION_MESSAGE);}
```

Titulo del diálogo

ALGUNAS COSITAS..

JOPtionPane

Mensaje de advertencia:




```
JOptionPane.showMessageDialog(this,
                    "Debe deligenciar los nombres de todos Jurados",
                    "Advertencias",
                    JOptionPane.INFORMATION_MESSAGE);}
```

Ícono y botones del diálogo

ALGUNAS COSITAS..

JOPtionPane

Mensaje de confirmación:


```
int respuesta =
        JOptionPane.showConfirmDialog(this,
        "Esta seguro de Instalar Mesa de Votacion?",
        "Confirmacion",
         JOptionPane.YES_NO_OPTION);
```

ALGUNAS COSITAS..

JOPtionPane

Mensaje de confirmación:

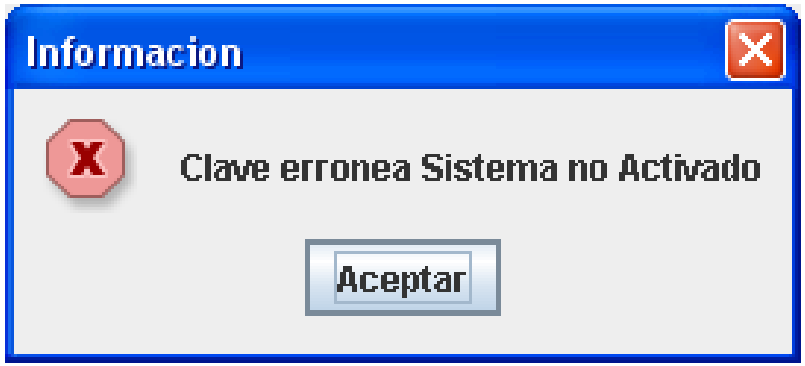

```
int respuesta =
        JOptionPane.showConfirmDialog(this,
        "Esta seguro de Instalar Mesa de Votacion?",
        "Confirmacion",
         JOptionPane.YES_NO_OPTION);
```

Devuelve un valor de tipo entero

ALGUNAS COSITAS..

JOPtionPane

Mensaje de Error:




```
JOptionPane.showMessageDialog(this,
                    "Clave erronea Sistema no Activado",
                    "Informacion",
                    JOptionPane.ERROR_MESSAGE);
```

ALGUNAS COSITAS..

JOPtionPane

Mensaje de con ícono personalizado:


```
ImageIcon icono = new ImageIcon(getClass().getResource("colibri_p.png"));

JOptionPane.showMessageDialog(null, "¡Adios!","Finalización",
                                    JOptionPane.QUESTION_MESSAGE, icono);
```

ALGUNAS COSITAS..

Personalizando JOptionPane

```
imagen = new ImageIcon(getClass().getResource("perrito.GIF"));

UIManager UI=new UIManager();
//cambiar Apariencia de JOptionPane,
UI.put("OptionPane.background", new Color(175, 50, 10));
UI.put("Panel.background", Color.white);
//nombre, estilo, tamaño de fuente
UI.put("OptionPane.messageFont", new Font("Tempus Sans ITC", 3, 15));
UI.put("OptionPane.messageForeground", Color.blue);

JOptionPane.showMessageDialog(null, "Adios!!", "Titulo de Joption", 1, imagen);
```

MANUALES

- https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html
- http://superoptimo.wdfiles.com/local--files/programacion-java/Manual_Java_Swing.pdf
- http://www.javabeginner.com/java-swing/java-swing-tutorial
- http://zetcode.com/tutorials/javaswingtutorial/
- http://www.academica.mx/sites/default/files/adjuntos/13274/Como%20programar%20en%20Java%20-%207ma%20Edicion%20-%20P.%20J.%20Deitel.pdf

REFERENCIAS

- JAVA Como Programar séptima edición DEITEL. Pearson.
- JAVA 2 cuarta edición. McGraw Hill 2003
- http://docs.oracle.com/javase/1.5.0/docs/guide/swing/
- http://es.wikipedia.org/wiki/Swing_%28biblioteca_gr%C3%A1fica%29
- http://www.java2s.com/Tutorial/Java/0240__Swing/CustomizingaJOptionPaneLookandFeel.htm