







## JAVA [SWING BIBLIOTECA GRÁFICA] Parte 2.

### COMPONENTES DE ALTO NIVEL PARA VENTANAS

 JFrame	 JDialog	 JApplet
 JWindow	 JInternalFrame	







### COMPONENTES DE TEXTO

 JTextField	 JTextPane	 JPasswordField
 JEditorPane	 JTextArea	 FormattedTextField







### COMPONENTES DE MENU

 JMenuBar	 JMenu	 JCheckboxMenuItem
 JPopupMenu	 JMenuItem	 JRadioButtonMenuItem













### CONTENEDOR DE COMPONENTES

 JMenuBar	 JMenu	 JCheckboxMenuItem
 JPopupMenu	 JMenuItem	 JRadioButtonMenuItem

### COMPONENTES DE COMPLEJO

 JTable	 JTree	 JFileChooser
 JList	 JOptionPane	 JColorChooser

### COMPLEJOS ATOMICOS

 JLabel	 JButton	 JToggleButton
 JCheckBox	 JRadioButton	 JProgressBar
 JSpinner	 JComboBox	 JSlider
 JScrollBar	 JSeparator	 JToolTip

### Componentes Swing

=====

#### General:

setToolTipText(): muestra mensaje cuando se coloca el cursor

SwingConstants : declara un conjunto de constantes enteras para usar con los componentes swing

getSource(): devuelve una referencia del origen del evento

getActionCommand(): devuelve el texto que hay en un JLabel, JButton o en un JTextField

#### Imágenes:

ImageIcon(): soporta varios formatos de imagen

getClas(): obtiene una referencia al objeto Class que representa la clase para el objeto que llama al método

getResource(): devuelve la ubicación de su argumento en forma url.

#### JLabel:

setText(): Le coloca el String que recibe como argumento

getText(): Obtiene el texto que tiene el componente

setIcon(): especifica un objeto Icon a mostrar en un JLabel

getIcon(): obtiene el objeto Icon que esta en un JLabel

setHorizontalAignment(): alinea un JLabel horizontalmente

setVerticalAlignment(): alinea un JLabel Verticalmente

#### JButton :

setText(): Le coloca el String que recibe como argumento

getText(): Obtiene el texto que tiene el componente

setIcon(): especifica un objeto Icon a mostrar en un JLabel

getIcon(): obtiene el objeto Icon que esta en un JLabel  
setRolloverIcon(): imagen a mostrar en el JButton cuando se coloca el cursor sobre el  
ActionListener: Interface de escucha que permite vigilar a un JButton en caso haya una accion sobre el.  
ActionEvent: Cuando ActionListener detecta que hay una accion sobre el, envia un objeto ActionEvent.  
actionPerformed(): Este metodo recibe el objeto ActionEvent y en el haremos las acciones que queramos.

### **JTextField:**

setText(): Le coloca el String que recibe como argumento  
getText(): Obtiene el texto que tiene el componente  
ActionListener: Interface de escucha que permite vigilar a un JButton en caso haya una accion sobre el.  
ActionEvent: Cuando ActionListener detecta que hay una accion sobre el, envia un objeto ActionEvent.  
actionPerformed(): Este metodo recibe el objeto ActionEvent y en el haremos las acciones que queramos.

### **JPasswordField:**

setText(): Le coloca el String que recibe como argumento  
getPassword(): Obtiene el password que tiene el componente.  
ActionListener: Interface de escucha que permite vigilar a un JButton en caso haya una accion sobre el.  
ActionEvent: Cuando ActionListener detecta que hay una accion sobre el, envia un objeto ActionEvent.  
actionPerformed(): Este metodo recibe el objeto ActionEvent y en el haremos las acciones que queramos.

### **JCheckBox:**

setMaximunRowCount(): establece el maximo de elementos a mostrar en un combo  
isSelected(): determina si un objeto JCheckBox esta seleccionado.  
getSelectedItem(): devuelve el objeto seleccionado  
getSelectionIndex(): devuelve la posicion del objeto seleccionado  
setSelectedItem(): establece el objeto seleccionado  
setSelectionIndex(): establece el obeto seleccionado por medio de indice  
ItemLisemter: Interface de escucha de un JComboBox, JList.  
ItemEvent: Cuando se detecta accion en un JComboBox, JList, etc; se envia un evento ItemEvent.  
itemStateChanged(): Método encargado de desencadenar acciones en un evento tipo ItemEvent.

### **JRadioButton:**

ItemEvent

- Se tiene que agrupar con un ButtonGroup (javax.swing)

### **JList:**

ListSelectionEvent, ListSelectionListener

- Debe implementar el método valueChanged()
- setVisibleRowCount(): muestra la cantidad visible de elementos  
setSelectionModel(): especifica el modelo de selección de una JList  
setListData(): elementos a mostrar en el JList  
getSelectedValues(): devuelvo un arreglo Object con los elementos seleccionados  
setFixedCellWidth(): establece la anchura de un objeto JList.  
setFixedCellHeight(): establece la altura de cada elemento en un objeto JList.

## **JTextArea**

- Un objeto JTextArea proporciona un área para manipular varias líneas de texto. JTextArea es una subclase de JTextComponent, la cual declara métodos comunes para objetos JTextField, JTextArea y varios otros componentes de GUI basados en texto.
  - La clase Box es una subclase de Container que utiliza un administrador de esquemas BoxLayout para ordenar los componentes de la GUI, ya sea en forma horizontal o vertical.
  - El método static createHorizontalBox de Box crea un objeto Box que ordena los componentes de izquierda a derecha, en el orden en el que se adjuntan.
  - El método getSelectedText (que hereda JTextArea de JTextComponent) devuelve el texto seleccionado de un objeto JTextArea.
  - Podemos establecer las políticas de las barras de desplazamiento horizontal y vertical de un objeto JScrollPane momento de crearlo. Los métodos setHorizontalScrollBarPolicy y setVerticalScrollBarPolicy de JScrollPane pueden usarse para modificar las políticas de las barras de desplazamiento en cualquier momento.
- setWrapStyleWord(): establece el salto de línea entre palabras cuando el texto ocupe el ancho del área

## **Eventos de mouse**

=====

- Las interfaces MouseListener y MouseMotionListener se utilizan para escuchar los eventos de mouse.
  - La interface MouseInputListener extiende a MouseListener y MouseMotionListener.
  - Cada uno de los métodos manejadores de eventos del ratón recibe un objeto MouseEvent como argumento. Un objeto MouseEvent contiene información acerca del evento de ratón que ocurrió, incluyendo las coordenadas x y y de la ubicación en donde ocurrió el evento. Estas coordenadas se miden empezando desde la esquina superior izquierda del componente de la GUI en donde ocurrió el evento.
  - La interfaz MouseWheelListener permite a las aplicaciones responder a la rotación de la rueda de un ratón.
  - Los componentes de la GUI heredan los métodos addMouseListener y addMouseMotionListener de la clase Component.
  - Podemos extender una clase adaptadora para que herede la implementación predeterminada de cada método, y por consiguiente, podemos sobrescribir sólo el (los) método(s) necesario(s) para el manejo de eventos.
  - El método getClickCount de MouseEvent devuelve el número de clics de los botones del ratón.
  - Los métodos isMetaDown e isAltDown determinan cuál botón del ratón oprimió el usuario.
- getX(): devuelve el punto x sobre el que está el mouse  
getY(): devuelve el punto y sobre el que está el mouse  
isControlDown(): determina si se ha presionado la tecla Ctrl.

## **Eventos de tecla**

=====

La interfaz KeyListener se utiliza para manejar eventos de teclas, que se generan cuando se oprimen y sueltan las teclas en el teclado. El método addKeyListener de la clase Component registra un objeto KeyListener para un componente.

- El método `getKeyCode` de `KeyEvent` obtiene el código de tecla virtual de la tecla oprimida. La clase `KeyEvent`

mantiene un conjunto de constantes de código de tecla virtual que representa a todas las teclas en el teclado.

- El método `getKeyText` de `KeyEvent` devuelve una cadena que contiene el nombre de la tecla que se oprimió.
- El método `getKeyChar` de `KeyEvent` obtiene el valor Unicode del carácter escrito.
- El método `isActionKey` de `KeyEvent` determina si la tecla en un evento fue una tecla de acción.
- El método `getModifiers` de `InputEvent` determina si se oprimió alguna tecla modificadora (como Mayús, Alt y

Ctrl ) cuando ocurrió el evento de tecla.

- El método `getKeyModifiersText` de `KeyEvent` produce una cadena que contiene los nombres de las teclas modificadoras que se oprimieron.

### **PaintComponent() y Clase Graphics:**

=====

- Los componentes ligeros de Swing que extienden a la clase `JComponent` contienen el método `paintComponent`, el cual se llama cuando se muestra un componente ligero de Swing. Al sobrescribir este método, puede especificar cómo dibujar figuras usando las herramientas de gráficos de Java.
- Al personalizar un objeto `JPanel` para usarlo como un área dedicada de dibujo, la subclase debe sobrescribir el método `paintComponent` y llamar a la versión de `paintComponent` de la superclase como la primera instrucción en el cuerpo del método sobrescrito.
- Las subclases de `JComponent` soportan la transparencia. Cuando un componente es opaco, `paintComponent` borra el fondo del componente antes de mostrarlo en pantalla.
- La transparencia de un componente ligero de Swing puede establecerse con el método `setOpaque` (un argumento `false` indica que el componente es transparente).
- La clase `Point` (paquete `java.awt`) representa una coordenada x-y.
- La clase `Graphics` se utiliza para dibujar.
- El método `getPoint` de `MouseEvent` obtiene el objeto `Point` en donde ocurrió un evento de ratón.
- El método `repaint` (heredado directamente de la clase `Component`) indica que un componente debe actualizarse en la pantalla lo más pronto posible.
- El método `paintComponent` recibe un parámetro `Graphics`, y se llama de manera automática cada vez que un componente ligero necesita mostrarse en la pantalla.
- El método `fillOval` de `Graphics` dibuja un óvalo relleno. Los cuatro parámetros del método representan el cuadro delimitador en el cual se muestra el óvalo. Los primeros dos parámetros son la coordenada x superior izquierda y la coordenada y superior izquierda del área rectangular. Las últimas dos coordenadas representan la anchura y la altura del área rectangular.

### **Layouts**

=====

- Los administradores de esquemas ordenan los componentes de la GUI en un contenedor, para fines de presentación.
- Todos los administradores de esquemas implementan la interfaz `LayoutManager` (paquete `java.awt`).
- El método `setLayout` de la clase `Container` especifica el esquema de un contenedor.
- `FlowLayout` es el administrador de esquemas más simple. Los componentes de la GUI se colocan en un contenedor, de izquierda a derecha, en el orden en el que se agregaron al contenedor. Cuando se llega al borde del contenedor, los componentes siguen mostrándose en la siguiente línea. La clase `FlowLayout` permite a los componentes de la GUI alinearse a la izquierda, al centro (el valor predeterminado) y a la derecha.

- El método `setAlignment` de `FlowLayout` cambia la alineación para un objeto `FlowLayout`.
- El administrador de esquemas `BorderLayout` (el predeterminado para un objeto `JFrame`) ordena los componentes en cinco regiones: `NORTH`, `SOUTH`, `EAST`, `WEST` y `CENTER`.
- Un `BorderLayout` limita a un objeto `Container` para que contenga cuando mucho cinco componentes; uno en cada región.
- El administrador de esquemas `GridLayout` divide el contenedor en una cuadrícula, de manera que los componentes puedan colocarse en filas y columnas.
- El método `validate` de `Container` recalcula el esquema del contenedor, con base en el administrador de esquemas actual para ese objeto `Container` y el conjunto actual de componentes de la GUI que se muestran en pantalla.

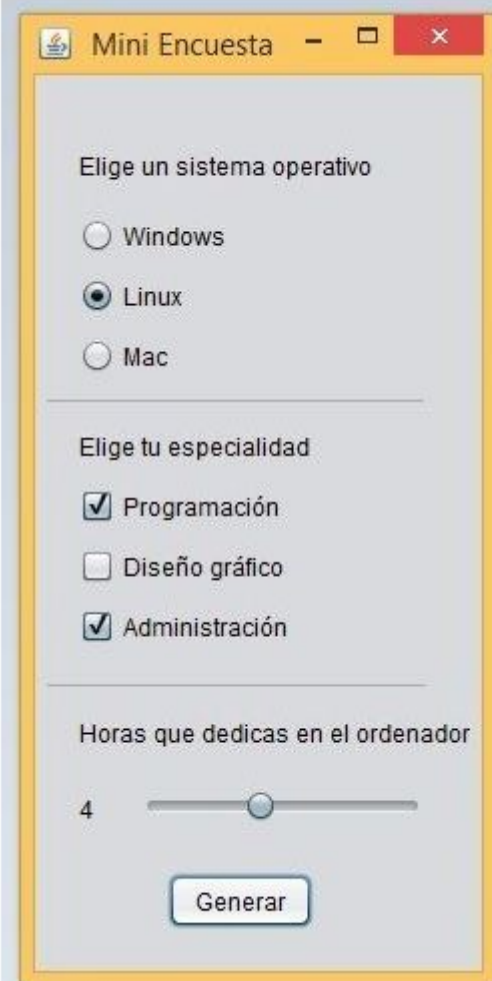
## Layouts

A cada contenedor se le establece un *layout* asociado

Un *layout* establece la disposición de los componentes dentro del contenedor

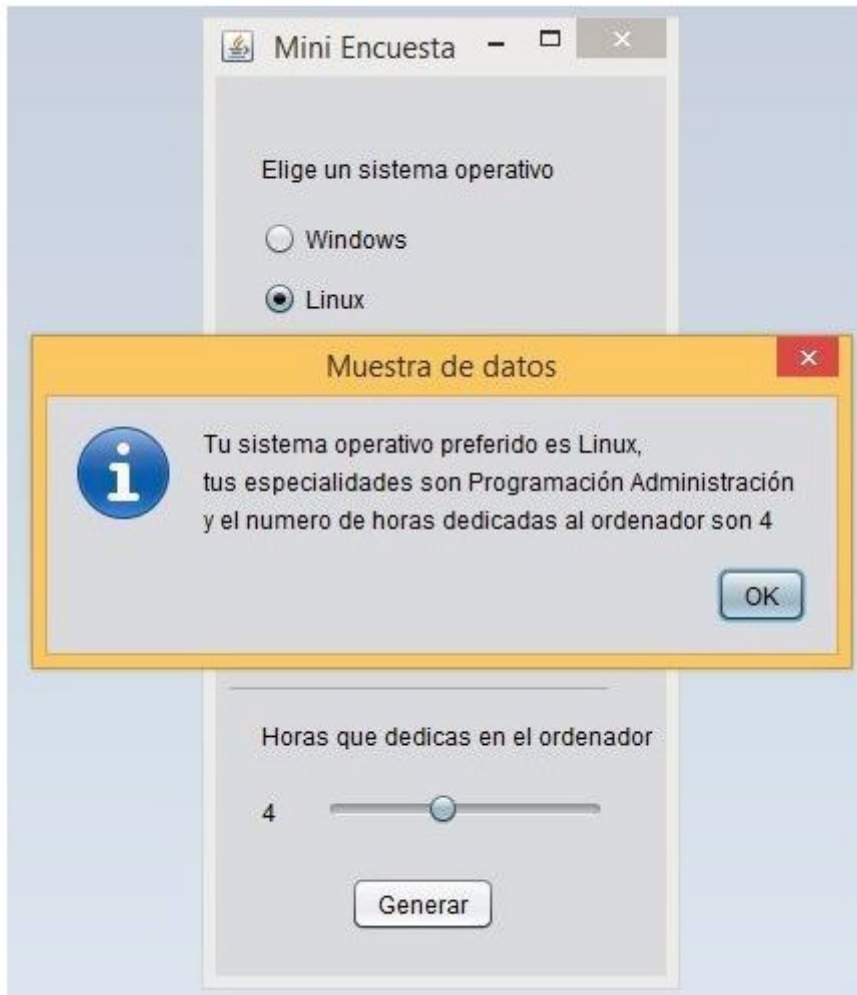
Los componentes se añaden a un contenedor con el método `add`

### 1- Ejercicio:



The image shows a Java Swing window titled "Mini Encuesta" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a survey form with the following elements:

- Elige un sistema operativo**: Three radio buttons for "Windows", "Linux" (selected), and "Mac".
- Elige tu especialidad**: Three checkboxes for "Programación" (checked), "Diseño gráfico" (unchecked), and "Administración" (checked).
- Horas que dedicas en el ordenador**: A slider control with a value of 4 displayed on the left.
- Generar**: A button at the bottom of the form.



Crea una minienquesta gráfica. Daremos una serie de opciones para que el usuario elija. La encuesta preguntará lo siguiente:

- Elige un sistema operativo (solo una opción, JRadioButton)
  - Windows
  - Linux
  - Mac
- Elige tu especialidad (pueden seleccionar ninguna o varias opciones, JCheckBox)
  - Programación
  - Diseño gráfico
  - Administración
- Horas dedicadas en el ordenador (usaremos un slider entre 0 y 10)

Para el slider, os recomiendo usar un JLabel, que os diga que valor tiene el slider, usad el evento `stateChanged`.

//este el codigo necesario para dar accion a un boton

```
JButton unBoton = new JButton("Click");
```

```
unBoton.addActionListener(new ActionListener(){
```

```
    public void actionPerformed(ActionEvent ae){
```

```
        //aqui pones lo que quieras hacer al presionar el boton
```

```
        //Yo pondre un aviso que diga Presionado
```

```
        JOptionPane.showMessageDialog(null, "Presionado");
```

```
    }
```

```
});
```

2) Crea una simple lista de peliculas. tendremos un JComboBox, donde almacenaremos las peliculas, que vayamos almacenando en un campo de texto. Al pulsar el botón **Añadir** la pelicula que hayamos metido, se introducirá en el JComboBox.

