**Academia Java – JAVA**

***Usando Swing a tu Ritmo***

Tabla de contenido

[TRABAJANDO EN TU SWING 3](#_Toc314213003)

[1. Componentes de Swing 4](#_Toc314213004)

[1.1. Los componentes pueden anidarse 4](#_Toc314213005)

[1.2. Cuatro pasos para crear un GUI (repaso) 4](#_Toc314213006)

[2. Layout Managers 5](#_Toc314213007)

[3. ¿Cómo deciden los layout managers? 6](#_Toc314213008)

[3.1. Diferentes layout managers tienen diferentes políticas 7](#_Toc314213009)

[3.2. Los tres grandes: border, flow y box 8](#_Toc314213010)

[3.2.1. BorderLayout cuida de 5 regiones: este oeste, norte, sur y centro 9](#_Toc314213011)

[3.2.2. FlowLayout se preocupa por el flujo de los componentes: de izquierda a derecha y de arriba a abajo, en el orden en que se han añadido 13](#_Toc314213012)

[3.2.3. BoxLayout al rescate! Mantiene los componentes apilados, aunque haya espacio para ponerlos uno al lado de otro 17](#_Toc314213013)

[3.3. Jugando con los componentes de Swing 20](#_Toc314213014)

[3.3.1. JTextField 20](#_Toc314213015)

# TRABAJANDO EN TU SWING

¡Swing es fácil! El código de esta librería se ve fácil, pero después cuando lo compilas, lo ejecutas, lo ves y piensas “¿eso deberia estar ahí?”. Lo mismo que lo hace fácil, es lo mismo que lo hace algo especial para controlarlo –me refiero al Layout Manager. Los objetos de este controlan el tamaño y la localización de los widgets en una GUI JAVA. Ellos van a hacer montones de trabajo a tu favor, pero no siempre te van a gustar los resultados. Si deseas dos botones exactamente del mismo tamaño, no lo son. Si quieres que los campos de ingreso de texto, text fields, solo sean de tres pulgadas de logitud, son de nueve. O la etiqueta está abajo cuando deberia estar al lado.

Si, parece tedioso…pero con un pequeño trabajo puedes lograr controlar los layout managers. En esta parte, vamos a trabajar con Swing y sus layout managers, y aprenderemos más acerca de estos widgets

# Componentes de Swing

Componente es un término más adecuado para nombrar a lo que nosotros llamamos widget. Ósea, lo que colocas en un GUI. Las cosas que van a interactuar con el usuario. Las cajas de texto, buttons, listas con scroll, radio buttons, etc, todos ellos son componentes. De hecho, ellos extienden de javax.swing.JComponent.

# Los componentes pueden anidarse

En Swing, virtualmente todos los componente son capaces de tomar otros componentes. En otras palabras, puedes algo en cualquier cosa. Pero la mayor parte del tiempo, agregarás componentes interactivos como botones y listas en componentes *background* como frames y paneles. Aunque es posible poner un panel dentro de un boton, seria raro y no te haria ganar ningún premio a la usabilidad de la aplicación.

Con excepción del JFrame, las diferencias entre componentes *interactivos* y componentes *background* son artificiales. Un JPanel, por ejemplo, es frecuentemente usado como background para agrupar otros componentes, pero un JPanel puede ser interactivo, pues al igual que con otros componentes puede registrarle eventos tales como los click del mouse y otros.

# Cuatro pasos para crear un GUI (repaso)

|  |  |
| --- | --- |
|  | Crear una ventana (un JFrame) |
|  | JFame frame = new JFrame(); |
|  | Crea un componente (button, text field, etc.) |
|  | JButton button = new JButton(“click me”); |
|  | Agrega el componente al frame |
|  | frame.getContentPane().add(BorderLayout.EAST, button); |
|  | Muestralo (dale un tamaño y hazlo visible) |
|  | frame.setSize(300, 300);  frame.setVisible(true); |

|  |  |
| --- | --- |
| **Componentes interactivos** | **Componentes background** |
|  |  |

# Layout Managers

Un Layout Manager es un objeto Java asociado con un componente en particular, casi siempre con un componente *backgroud*.

El Layout Manager controla los componentes contenidos dentro del componente con el cual está asociado. Es decir, si un frame contiene un panel, y el panel contiene un button, el layout manager del panel controla el tamaño y posición del botón, mientras que el layout manager del frame controla el tamaño y posición del panel. Por otro lado, el botón no necesita un layout manager porque no puede contener otros componentes.

Si un panel contiene cinco cosas, incluso si cada una de esas cinco cosas tienen su propio layout manager, el tamaño y la posición de esas cinco cosas en el panel son controlados por el layout manager del panel. Si esas cinco cosas, en cambio, contiene otras cosas, entonces esas otras cosas son colocadas de acuerdo al layout manager de lo que las contenga.

Cuando decimos que toman un componente, realmente estamos diciendo que lo agrega dentro suyo. Es decir, un panel toma un botón porque el botón fue agregado al panel usando algo como:

myPanel.add(button);

Los layout managers vienen en diferentes sabores, y cada background component puede tener su propio layout manager. Los cuales siguen sus propias políticas para construir un layout. Por ejemplo, un layout manager puede determinar que todos los componentes en el panel sean del mismo tamaño y alineados a la matricula. Mientras que otro layout manager puede permitir que cada componente elija su propio tamaño, pero que estén ordenados verticalmente.

Algunos ejemplos:

JPanel panelA = new JPanel() ;

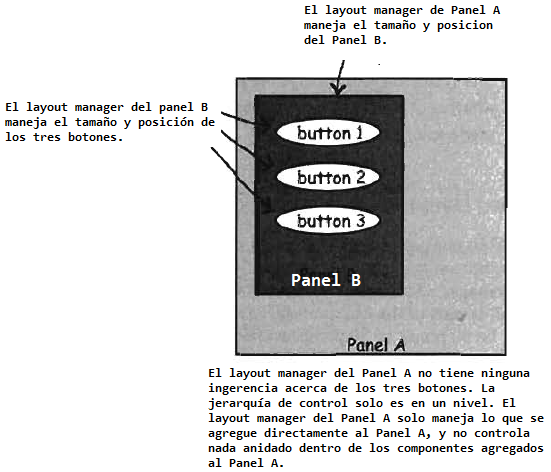
JPanel panelB = new JPanel() ;

panelB.add(new JButton("button 1");

panelB.add(new JButton("button 2");

panelB.add(new JButton("button 3"));

panelA.add(panelB);



# ¿Cómo deciden los layout managers?

Cada layout manager tiene sus propias politicas para organizar sus componentes. Pero en lineas generales el proceso para organizar los background component es de la siguiente forma:

|  |  |
| --- | --- |
|  | Crear un panel y agregarle tres botones. |
|  | El layout manager del panel pide a cada botón cual es el tamaño que prefiere. |
|  | El layout manager usa sus politicas para decidir si respeta todas, parte o ninguna de las preferencias de los botones. |
|  | Agregar el panel a un frame. |
|  | El layout manager del frame pide al panel su preferencia en cuanto a su tamaño. |
|  | El layout manager del frame usa sus politicas para decidir si respectar todo, parte o ninguna de las preferencias del panel. |



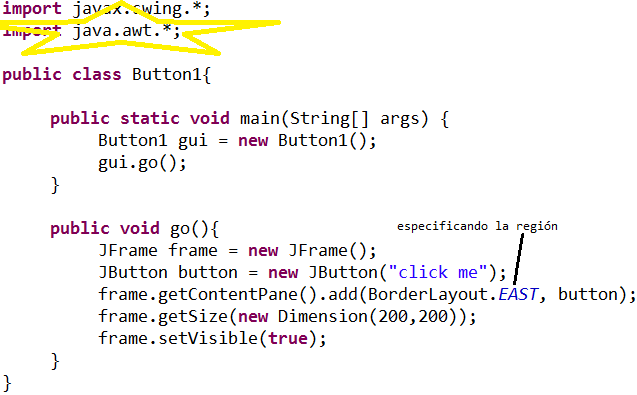
# Diferentes layout managers tienen diferentes políticas

Algunos layout managers respetan el tamaño de que los componentes le indican. Otro solo respetan una parte. Otros los del componente más largo y condiciona a este tamaño a los demás componentes. Y asi, se pueden desarrollar comportamientos más complejos, que con un poco de práctica aprenderás a controlar.

# Los tres grandes: border, flow y box

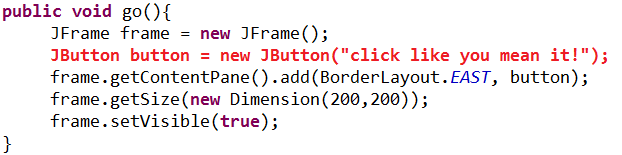
|  |  |
| --- | --- |
| **BorderLayout** |  |
| Divide al background component en cinco regiones. Puedes agregar solo un componente por region a componente controlado por un BorderLayout manager. Los componentes que se organizar en este manager generalmente no llegar a establecer sus preferencias en cuanto a tamaño. Este layout manager es el usado por defecto en un frame. |  |
| **FlowLayout** |  |
| Este manager actua como un procesador de texto, pero en vez de palabras componentes. Es decir, cada componente es del tamaño que tiene configurado y se organizan de izquierda a derecha en el orden que se van agregando, cuando la modalidad “word-wrap” está encendida. Entonces cuando un componente no calza horizontalmente, se coloca en la siguiente “linea” del alyout. Es el layout manager usado por defecto en un panel. |  |
| **BoxLayout** |  |
| Es como el FlowLayout pues cada componente tiene su propio tamaño y están ordenados en el orden en que fueron agregados. Pero, la diferencias es que este apila los componentes verticalmente (o horizontalmente, pero usualmente es vertical). Pero a diferencia del FlowLayout, no tiene el “word-wrapping atomático” y en su lugar se agrega un tipo de “component return key” que fuerza al componente a empezar una “nueva linea”. |  |

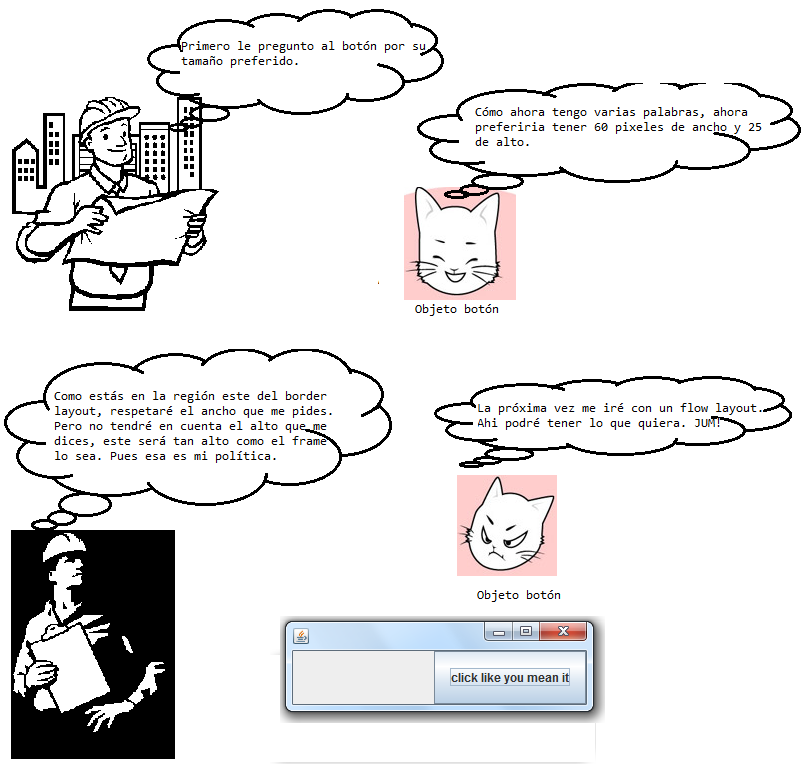
# BorderLayout toma 5 regiones: este oeste, norte, sur y centro



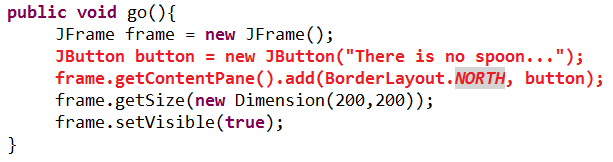


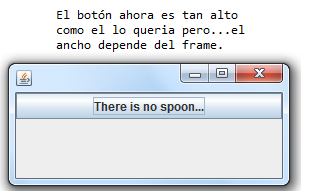
**Mira que pasa cuaando el damos más letras al botón…**



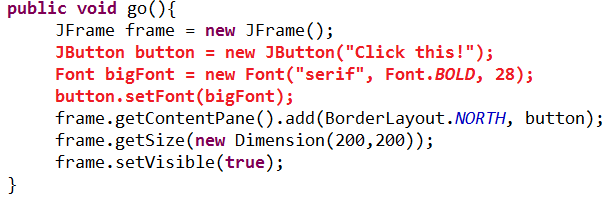


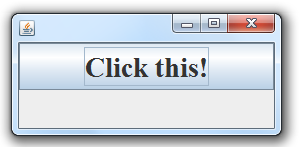
**Ahora en el norte…**

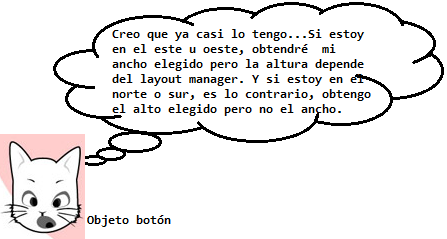




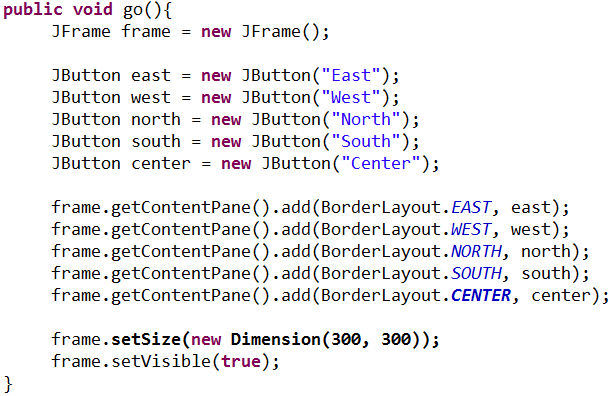
**Ahora haremos al botón un poco más alto!**

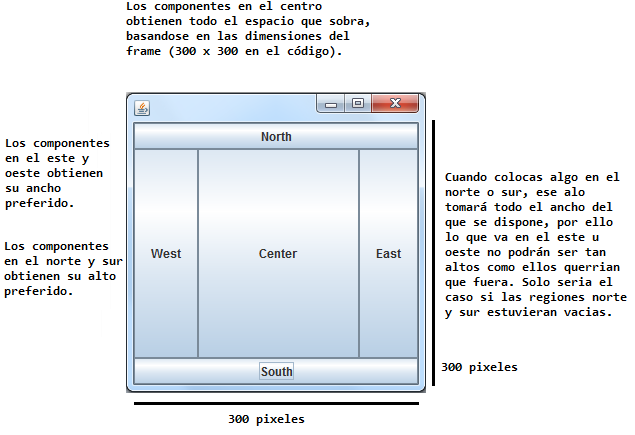






**¿Pero que pasa en la región del centro?**

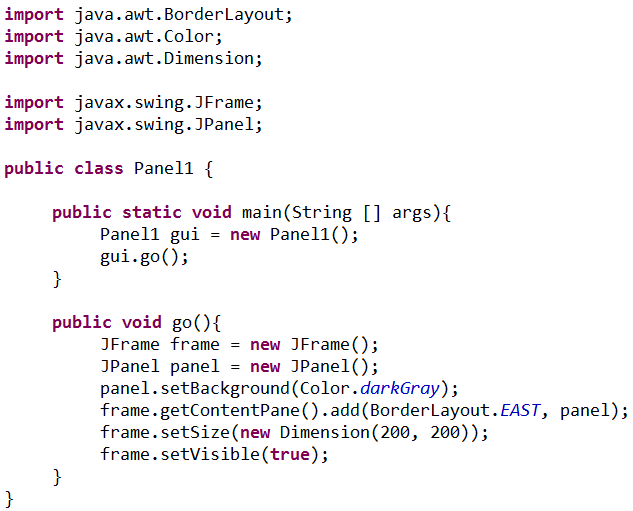


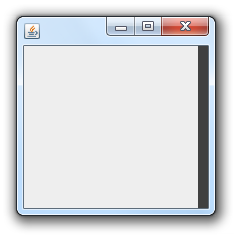


# FlowLayout se preocupa por el flujo de los componentes: de izquierda a derecha y de arriba a abajo, en el orden en que se han añadido

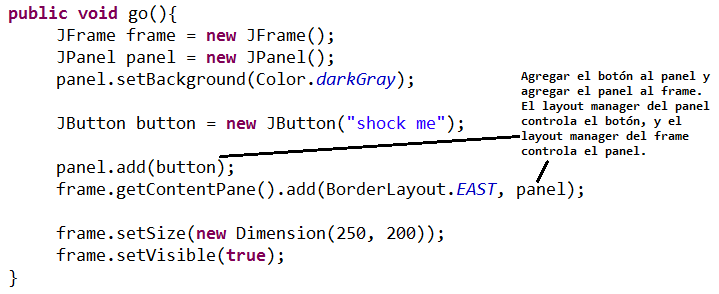
Vamos a agregar un panel a la región este.

Ya sabemos que el FlowLayout es el layout manager del JPanel. Entonces cuando agregamos un panel a un frame, el tamaño y la posición del panel está aun bajo el control del BorderLayout manager. Pero cualquier cosa dentro del panel (es decir, cualquier cosa agregada con panel.add(aComponent) esta bajo el control del FlowLayout manager. Empezaremos colocando un panel vacio en la región este del frame, y en las siguientes páginas agregaremos más cosas al panel.



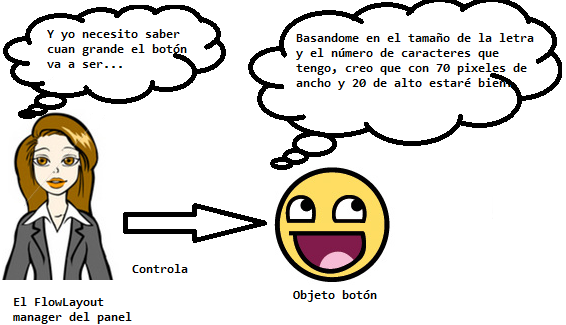


**Agreguemos un botón al panel.**

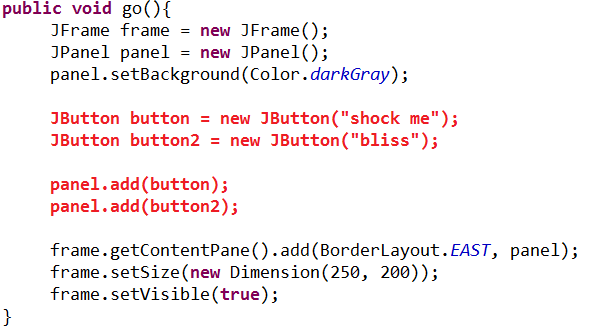


|  |  |
| --- | --- |
|  |  |





**¿Qué pasa si agregamos dos botones al panel?**

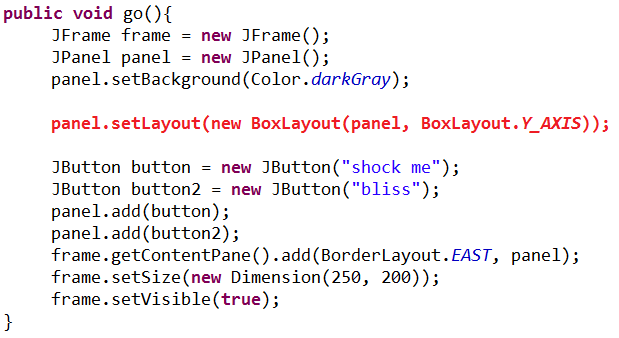


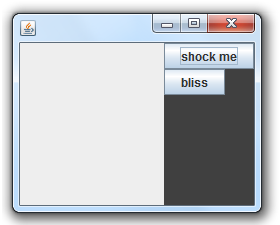
|  |  |
| --- | --- |
| **Lo que queremos…** | **Lo que obtenemos…** |
|  |  |

# ¡BoxLayout al rescate! Mantiene los componentes apilados, aunque haya espacio para ponerlos uno al lado de otro

A diferencia de sus otros dos primos, puede forzar una “nueva linea” para que los componentes se apilen en la siguiente linea, aunque haya espacio para que se ajusten horizontalmente.

Y ahora lo que debes hacer es modificar el layout manager por defecto del panel, es decir el FlowLayout, hacia un BoxLayout.



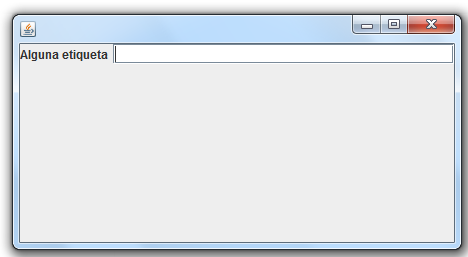


|  |  |
| --- | --- |
| * Los layout managers controlan el tamaño y la posición de los componentes anidados dentro de otros componentes. * Cuando agregas un componente a otro componente, el componente agregado es controlado por el layout manager del background component. * Un layout manager consulta a los componentes por su tamaño preferido, antes de tomar la decisión acerca del layout. Dependiendo de las políticas del layout podria respetar a todas, algunas o ninguna de las preferencias del los componentes. * El BoderLayout manager permite agregar un componente a una de sus cinco regiones. Debes especificar a cual cuando agregas un componente usando la sintaxis add(BorderLayout.EAST, panel); * Con el BorderLayout, los componentes en el norte y sur pueden obtener sus altos preferidos, pero no los anchos que están definidos para ellos. Los componentes en el este y oeste obtienen su ancho definido, pero no el alto. El componente en el centro usa lo que sobra, a menos que uses pack(). * Le método pack() usa los tamaños definidos en el componente del centro, luego determina el tamaño del frame usando el centro como punto de inicio. Así construye todo lo demás en base a eso. * El FlowLayout coloca los componentes de izquierda a derecha, de arriba hacia abajo, en el orden en que se han agregado, obligando a ingresar una nueva línea solo cuando el componente no se ajusta horizontalmente. * El FlowLayout le da a los componentes los componentes que ellos prefieren. * BoxLayout permite alinear los componentes apilados de forma vertical, aun si pueden acomodarse de lado a lado. Como el FlowLayout, el BoxLayout usa el tamaño definido del componente en ambas dimensiones. * El BorderLayout es el manager layout por defecto para un frame y el FlowLayout lo es para un panel. * Si lo deseas un panel puede usar otro layout usando el método setLayout(). | C:\Users\ingas\Desktop\Atencion2.png |

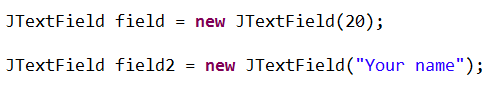
# Jugando con los componentes de Swing

Ya has aprendido lo básico de los layout managers, entonces ahora nos toca probar algunos de los componentes más comunes: text field, text area, checkbox y listas. No veremos todo el API, solo algunas cositas que deberias saber para usarlos.

# JTextField



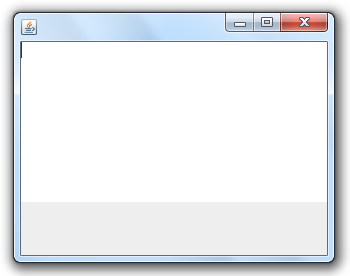
* **Constructores**



* **¿Cómo usarlo?**

|  |  |
| --- | --- |
|  | Obtener el texto  System.out.println(field.getText()); |
|  | Poner el texto  field.setText(“whatever”);  field.setText(“”); |
|  | Obtener un ActionEvent cuando el usuario lo apreta enter  field.addActionListener(myActionListener); |
|  | Seleccionar o resaltar el texto en el campo:  field.selectAll(); |
|  | Volver a colocar el cursor el campo, con lo cual el usuario vuelve a tipear:  field.requestFocus(); |

# JTextArea



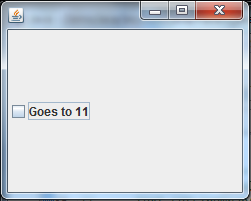
* **Constructores**

JTextArea text = **new** JTextArea(10,20);

* **¿Cómo usarlo?**

|  |  |
| --- | --- |
|  | Agregar scroll  JScrollPane scroller = new JScrollPane(text);  text.setLineWrap(true);  scroller.setVerticalScrollBarPolicy(  ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS);  scroller.setHorizontalScrollBarPolicy(  ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_ALWAYS);  frame.getContentPane().add(scroller); |
|  | Reemplazar el texto  text.setText(“Cualquier texto…”); |
|  | Agregar más texto al final  text.append(“ más texto ”); |
|  | Seleccionar o resaltar el texto en el campo:  text.selectAll(); |
|  | Colocar el cursor  text.requestFocus(); |

# JCheckBox



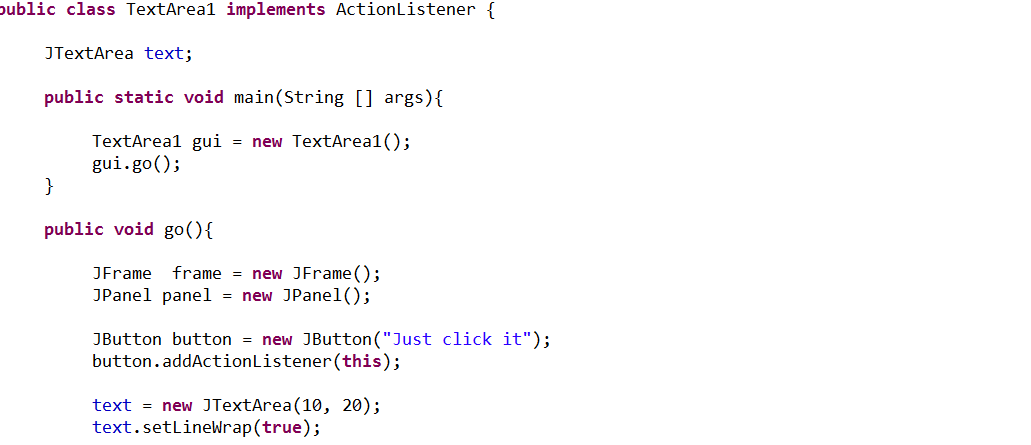
* **Constructores**

JCheckBox check = **new** JCheckBox("Goes to 11");

* **¿Cómo usarlo?**

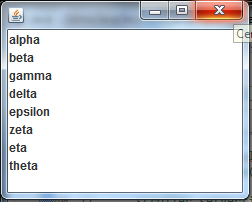
|  |  |
| --- | --- |
|  | Escuchar eventos  check.addItemListener(this); |
|  | Manejar el evento  public void itemStateChanged(){  String onOrOff = “off”;  if (check.isSelected()) onOrOff = “on”;  System.out.println(“Check box is ”+ onOrOff);  } |
|  | Seleccionar o deseleccionar via código:  check.setSelected(true);  check.setSelected(false); |

Un ejemplito:





# JList



* **Constructores**

String [] entries = {"alpha", "beta", "gamma", "delta", "epsilon", "zeta", "eta", "theta "};

JList list = new JList(entries);

* **¿Cómo usarlo?**

|  |  |
| --- | --- |
|  | Agregar scrollbar horizontal:  JScrollPane scroller = new JScrollPane(list);  scroller.setVerticalScrollBarPolicy(  ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS);  scroller.setHorizontalScrollBarPolicy(  ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_ALWAYS);  frame.getContentPane().add(scroller); |
|  | Colocar cantidad de lineas antes de aplicar el scroll  list.setVisibleRowCount(4); |
|  | Elegir solo un solo item por vez:  list.addListSelectionListener(this); |
|  |  |
|  |  |