**Conocimiento: AWT – Componentes**

Una interfaz gráfica está construida en base a varios widgets o elementos gráficos básicos denominados: los Componentes.

Ejemplos de estos Componentes son los **botones**, **barras de desplazamiento,** **etiquetas**, **listas**, **cajas de selección** o **campos de texto**. Los Componentes permiten al usuario interactuar con la aplicación y proporcionar información desde el programa al usuario sobre el estado del programa. En el AWT, todos los Componentes de la interface de usuario son instancias de la clase **Component** o uno de sus subtipos.

Los componentes también reciben el nombre de controles.

* Componentes:
  + - * Label (Etiqueta)
      * Button (Botones)
      * Chekbox (Casilla verificación)
      * Choice (Listas de opciones)
      * List (Listas)
      * ScrollBar (Barras de desplazamiento)
      * TextField, TextArea (Editores de texto)

**Clase Component**

La clase Component es la clase padre de toda la jerarquía de elementos gráficos definidos en AWT. Esta clase es abstracta y sirve como referencia para todos los widgets.

* Constructor
* Utilidades: LA clase Component hereda de Object y provee una serie de métodos que están disponibles para sus subclases, entre ellos tenemos:
  + - * **String getName ():** este método devuelve un String correspondiente al nombre del componente.
      * **void setName (String nombre)**: este método permite cambiar o asignar el nombre al componente.
      * **void setEnabled (boolean b):** este método permite activar o desactivar el componente correspondiente.
      * **void setVisible (boolean b**): este método permite mostrar o ocultar el componente correspondiente.
      * **Color getBackground ():** este método permite obtener el objeto Color correspondiente al color del fondo del componente.
      * **void setBackground(Color c):** este método permite cambiar o asignar el color del fondo del componente.
      * **Font getFont ():** este método devuelve el tipo de letra del componente.
      * **void setFont ():** Asigna el tipo de letra de dicha componente.
      * **Dimension getSize ():** este método permite obtener el objeto de la clase Dimension que corresponde al tamaño del componente.
      * **void setSize (int width,int height):** este método permite modificar el tamaño del componente.
      * **void setBounds (int x, int y, int width, int height):**  este método permite definir un una nueva posición y tamaño del componente.
      * **void paint (Graphics g):** este método permite dibujar en el componente.
      * **void update (Graphics g):** este método permite actualizar este componente.
      * **void repaint ():** este método permite volver a dibujar este componente.

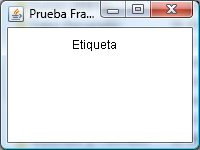
**Clase Label**

Esta clase hereda de Component, representa objetos conocidos como etiquetas, no permiten interacción con el usuario pero si permiten mostrar información en la gráfica que es de utilidad para poder comprender todo el contexto. Usualmente utilizamos una etiqueta para poner letreros o mensajes.

* Constructor:
  + **Label ():** este Constructor crea un objeto vacío.
  + **Label (String texto):** este constructor crea una nueva etiqueta con el String del parámetro de entrada, alineado a la izquierda.
  + **Label (String texto, int alineacion):** este constructor crea una nueva etiqueta con el String del parámetro de entrada y alineado según lo que indique el parámetro de tipo int.
* Utilidades: esta Clase hereda de Component, por tanto utiliza los métodos que están ahí definidos, pero además agrega nuevas utilidades, tales como:
  + **int getAlignment ():** este método retorna un valor de tipo int, correspondiente a la alineación es que se encuentra el texto dentro de la etiqueta, estos valores pueden ser: CENTER (que equivale al número 1), LEFT (que equivale al número 0) y RIGTH ( que equivale al número 2).
  + **String getText ():** este método retorna un String que representa el texto que contiene la etiqueta.
  + **void setAlignment (int alineacion):** este método permite cambiar la alineación del texto de la etiqueta según lo definido en el parámetro de entrada. Puede ser Label.LEFT, Label.RIGTH o Label.CENTER.
  + **void setText (String texto):** este método permite cambiar el texto de la etiqueta por lo especificado en el parámetro de entrada.

Ejemplo de uso de un Label en un contenedor

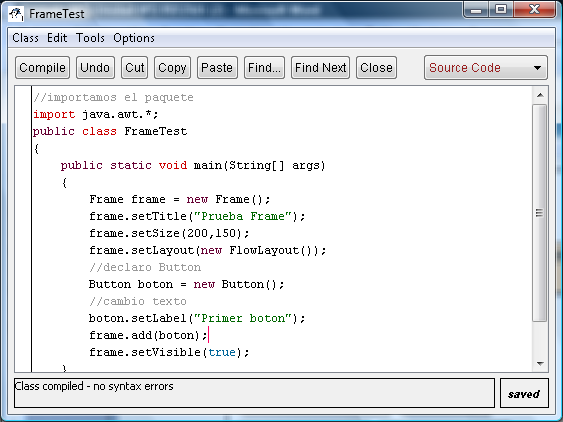
Al revisar el código anterior observamos que se declara un label, se le asigna un texto y se agrega al contenedor. El resultado de ejecutar el método main es:



**Clase Button**

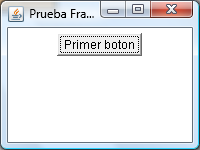
Esta clase hereda de la clase Component, construye un objeto de tipo botón, que contiene una etiqueta y que genera un evento cuando es pulsado o seleccionado. Este debe ser agregado a un contenedor para que sea visualizado.

* Constructor:
  + **Button ():** este método construye un objeto de la clase Button sin etiqueta.
  + **Button (String etiqueta):** este método construye un objeto de la clase Button con la etiqueta especificada en el parámetro de entrada.
* Utilidades: esta clase utiliza todos los métodos heredados de component, pero además agrega otras funcionalidades, tales como:
  + **void addActionListener (ActionListener l):** este método agrega un objeto que escucha el evento al presionar el botón y permite ejecutar un determinado número de acciones.
  + **ActionListener getActionListeners ():** este método retorna el objeto que escucha el evento en el botón.
  + **String getLabel ():** este método retorna un String conteniendo la etiqueta del botón.
  + **void setLabel (String etiqueta):** este método permite cambiar la etiqueta del botón con la del parámetro de entrada etiqueta.

Ejemplo de uso de un Button en un contenedor

En el ejemplo anterior podemos ver como se declara el boton, se le cambia el texto que mostrará y se agrega al contenedor para ser desplegado.

El resultado de la ejecución es:

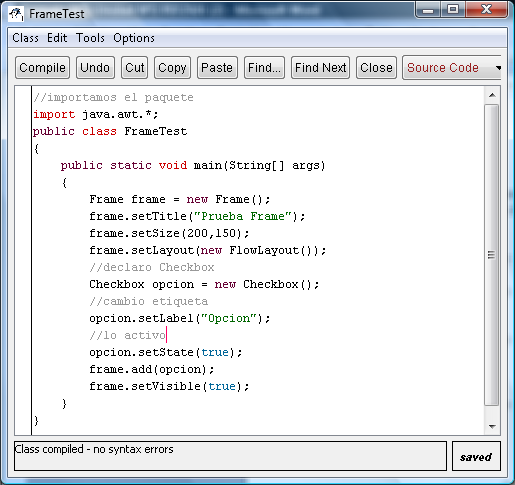


**Clase Checkbox**

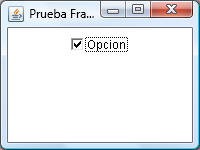
La clase Checkbox hereda de la clase Componente y proporciona objetos que constituyen cajas de chequeo, que contiene botones con dos estados que pueden estar "on" o "off".

* Constructor:
  + - * **Checkbox () :** este constructor crea un objeto Checkbox sin etiqueta y en estado off.
      * **Checkbox (String label):** este constructor crea un objeto Checkbox con la etiqueta definida en el parámetro de entrada y en estado off.
      * **Checkbox (String label, boolean estado):** este constructor crea un objeto Checkbox con la etiqueta definida en el parámetro de entrada y el estado definido por el parámetro “estado”.
      * **Checkbox (String label, boolean estado, CheckboxGroup grupo):** este constructor crea un objeto Checkbox con la etiqueta definida en el parámetro de entrada y el estado definido por el parámetro “estado” y asociado a un objeto CheckboxGroup.
      * **Checkbox(String label, CheckboxGroup grupo, boolean estado):** este constructor crea un objeto Checkbox con la etiqueta definida en el parámetro de entrada y el estado definido por el parámetro “estado” y asociado a un objeto CheckboxGroup.
    - Utilidades: esta clase utiliza todos los métodos heredados de la clase Component, además agrega algunas utilidades, tales como:
      * **void addItemListener (ItemListener l):** este método agrega un escuchante de acción cuando se active el Checkbox.
      * **CheckboxGroup getCheckboxGroup ():** este método permite obtener el CheckboxGroup al que pertenece.
      * **String getLabel ():** este método retorna la etiqueta asociada al Checkbox.
      * **boolean getState ():** este método permite obtener el estado del Checkbox, on u off.
      * **void removeItemListener (ItemListener l):** este método permite remover o eliminar el escuchante de evento.
      * **void setCheckboxGroup (CheckboxGroup g):** este método permite cambiar el CheckboxGroup al que está asociado el Checkbox.
      * **void setLabel (String label):** este método permite cambiar la etiqueta asociada al Checkbox.
      * **void setState (boolean estado):** este método permite activar o desactivar el Checkbox.

Ejemplo de uso de un Checkbox en un contenedor: En este ejemplo vemos como se declara, se configura y se agrega a un contenedor un Checkbox.



Al ejecutar este código el resultado es:



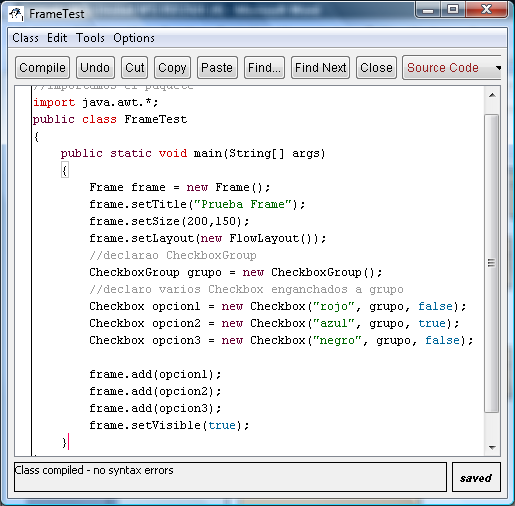
**Clase Checkboxgroup**

Esta clase hereda de Component y permite agrupar dos o más objetos Checkbox, de modo que solo uno de ellos esté activado (on)

Esta clase por sí sola no tiene sentido, deben existir Checkbox que se agreguen en él. No se agrega un objeto CheckboxGroup a un contendor sino que se hace a través del proceso de agregar Checkbox conectados con un determinado CheckboxGroup.

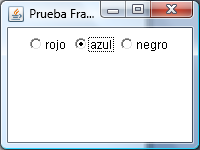
* Constructor:
  + - * **CheckboxGroup():** este constructor crea una nueva instancia de un CheckboxGroup vacío
* Utilidades: este clase hereda de Component y puede utilizar todos los métodos disponibles en dicha clases, además agrega otras funcionalidades tales como:
  + - * **Checkbox getSelectedCheckbox ():** este método devuelve el objeto Checkbox seleccionado.
      * **void setSelectedCheckbox (Checkbox box):** este método permite seleccionar un Checkbox en particular.

Ejemplo de uso de un CheckboxGroup en un contenedor:



En este ejemplo se declara un CheckboxGroup llamado grupo y se construyen tres Checkbox enganchados a grupo dentro del Constructor.

El resultado de ejecutar este código es el siguiente:

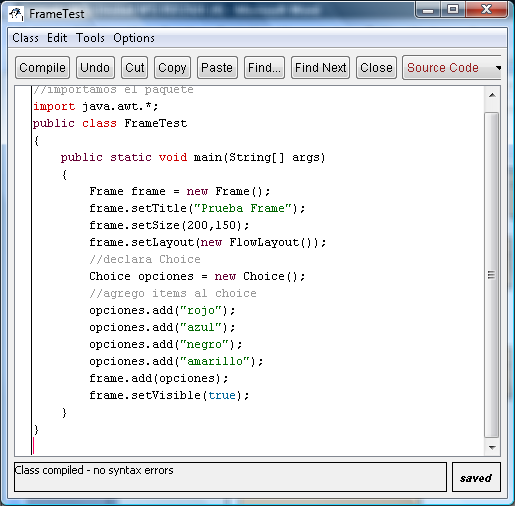


**Clase Choice**

Esta clase hereda de Component y representa objetos que son ventanas desplegables o pop-up que contiene varias opciones, dentro de la cuales se puede escoger alguna.

* Constructor:
  + - * **Choice ()**: este constructor crea un objeto Choice vacío.
  + Utilidades: esta clase hereda todos los métodos de la clase Component, además agrega otras utilidades tales como:
    - * **void add (String elemento):** este método permite agregar un elemento al final de la lista.
      * **void addItemListener (ItemListener listener):** este método permite agregar un objeto escuchante de los eventos de ítem del Choice.
      * **String getItem (int indice):** este método permite agregar un elemento o item en la posición especificada. El primer elemento está en la posición 0.
      * **int getItemCount():** este método retorna un valor entero correspondiente a la cantidad de elementos o ítems en el Choice
      * **int getSelectedIndex():** este método retorna la posición del elemento o item seleccionado.
      * **String getSelectedItem():** este método retorna el String correspondiente al elemento seleccionado.
      * **void insert (String item, int indice):** este método permite insertar el item indicado en el parámetro en la posición indice indicada en el parámetro.
      * **void remove (int posicion):** este método permite remover o eliminar el elemento en la posición indicada por el parámetro de entrada.
      * **void remove (String item):** este método permite remover o eliminar el elemento indicado en el parámetro de entrada.
      * **void select (int pos):** este método permite seleccionar o destacar el item de la posición indicada por el parámetro de entrada.
      * **void select (Strip item):** este método permite seleccionar o destacar el item indicado por el parámetro de entrada.

Ejemplo de uso de Choice en un contenedor:



En este ejemplo se crea un Choice llamado opciones, al que se le agregan varios item.

El resultado de ejecutar el código anterior es:

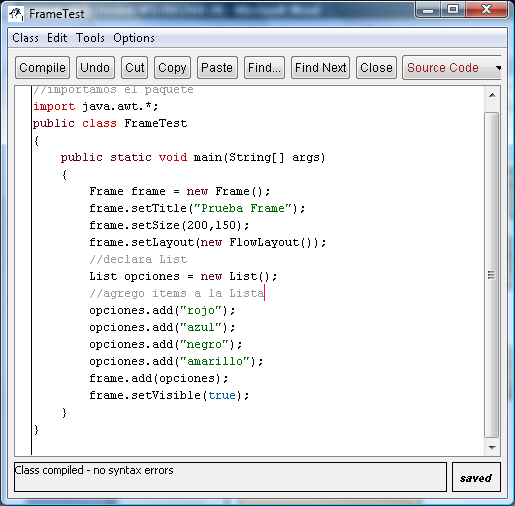


**Clase List**

Esta clase hereda de Component y representa una lista de elementos que son desplegados en una ventana desplazable y de los cuales se puede seleccionar uno o varios.

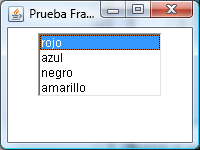
* + Constructor:
    - * **List ():** este constructor crea un nuevo objeto de tipo List, con cuatro líneas visibles, vacías por defecto y permitiendo selección única de elemento.
      * **List (int filas):** este constructor crea un nuevo objeto de tipo List, con las líneas visibles especificadas en el parámetro de entra, vacías por defecto y permitiendo selección única de elemento.
      * **List (int filas, boolean modo):** este constructor crea un nuevo objeto de tipo List, con las líneas visibles especificadas en el parámetro de entra, vacías por defecto y permitiendo selección simple o múltiple dependiendo del parámetro modo.
  + Utilidades: esta clase utiliza todos los métodos heredados de la clase Component, además agrega nuevas utilidades, tales como:
    - * **void add (String item**): este método permite agregar un item a la lista.
      * **void add (String item, int indice):** este método permite agregar un item a la lista en la posición indice.
      * **void addActionListener (ActionListener listener):** este método permite agregar un objeto escuchante de acción a la Lista.
      * **void addItemListener (ItemListener l):** este método permite agregar un objeto escuchante de item a la Lista.
      * **String getItem (int indice):** este método permite obtener el ítem que se encuentra en la posición indice.
      * **int getItemCount ():** este método devuelve un valor entero correspondiente a la cantidad de ítems de la lista.
      * **int getRows ():** este método devuelve un entero correspondiente al número de líneas que serán visibles a la vez en la lista.
      * **int getSelectedIndex ():** este método retorna un valor entero correspondiente al número de la posición del item seleccionado de la lista.
      * **String getSelectedItem ():** este método retorna un String contendido el item seleccionado de la lisa.
      * **void remove (int posición):** este método elimina el elemento de la lista en la posición indicada en el parámetro de entrada.
      * **void remove (String item):** este método permite eliminar el elemento item de la lista.
      * **void select (int indice):** este método permite seleccionar el elemento de la posición indice en la lista.

Ejemplo de uso de List en un contenedor:



En este ejemplo se crea un objeto de tipo List, se le agregan elementos y se agrega al contenedor.

El resultado de ejecutar el código anterior es:

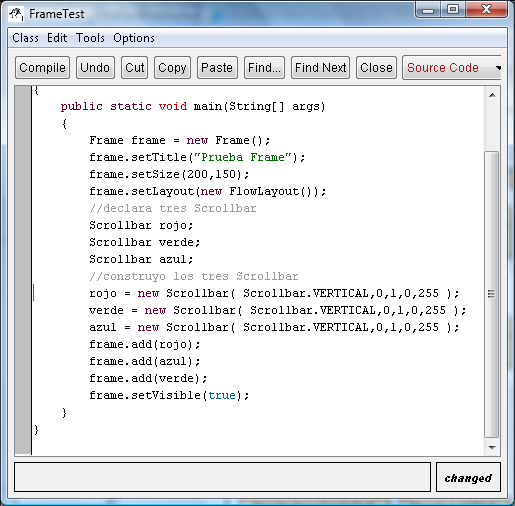


**Clase ScrollBar**

Esta clase hereda de Component y representa las barras de desplazamiento tanto vertical como horizontal. Una barra de desplazamiento puede actuar como un deslizador que el usuario manipula para seleccionar un valor

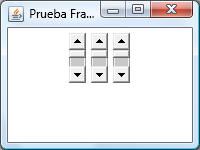
* + Constructor:
    - * **Scrollbar ():** este constructor crea una nueva barra de desplazamiento con orientación vertical.
      * **Scrollbar (int orientación):** este constructor crea una nueva barra de desplazamiento. El parámetro orientación indica si la barra de desplazamiento debe ser vertical u horizontal, esto especificado con Scrollbar.HORIZONTAL o Scrollbar.VERTICAL.
      * **Scrollbar (int orientación, int valorInicial, int tamaño, int mínimo, int máximo):** este constructor crea una nueva barra de desplazamiento. El parámetro orientación indica si la barra de desplazamiento debe ser vertical u horizontal, esto especificado con Scrollbar.HORIZONTAL o Scrollbar.VERTICAL, además valorInicial representa la ubicación inicial de la barra, tamaño representa el anchote la barra, mínimo y máximo representan los valores límites de la barra.
  + Utilidades: esta clase utiliza todos los métodos heredados de la clase Component, además agrega nuevas utilidades, tales como:
    - * **void addAdjustmentListener (AdjustmentListener listener):** este método permite agregar un objeto escuchante de ajuste a la barra de desplazamiento.
      * **int getValue ():** este método retorna un valor entero correspondiente al valor de la barra de desplazamiento.
      * **int getOrientation ():** este método retorna un valor entero correspondiente a la orientación de la barra de desplazamiento.
      * **void setOrientation (int orientacion):** este método permite cambiar la orientación de la barra de desplazamiento.
      * **void setValue (int nuevoValor):** este método permite cambiar el valor de la barra de desplazamiento.
      * **void setValues (int valor, int visible, int mínimo, int máximo):** este método permite cambiar el valor de la barra de desplazamiento, el tamaño de la barra, los valores mínimos y máximos de la barra.

Ejemplo de uso de Scrollbar en un contenedor:



En este ejemplo se declaran tres Scrollbar, se construyen y se agregan al contendor.

El resultado de la ejecución del código anterior es:



**Clase TextComponent**

Esta clase hereda de Component y representa la jerarquía de elementos gráficos que permiten manipular texto. No se instancia sino que se trabaja a través de sus clases hijas.

* + Utilidades: hereda todos los métodos de la clase Component, además agrega nuevas utilidades tales como:
    - * **Color getBackground ():** este método retorna el color de fondo del componente de texto.
      * **String getSelectedText ():** este método retorna un String conteniendo el texto seleccionado en el componente
      * **int getSelectionEnd():** este método retorna un entero correspondiente a la posición final del texto seleccionado en el componente
      * **int getSelectionStart():** este método retorna un entero correspondiente a la posición inicial del texto seleccionado en el componente
      * **String getText ():** este método retorna un String conteniendo el texto del componente.
      * **boolean isEditable ():** este método verifica si el texto puede o no ser editado.
      * **void setBackground (Color c):** este método permite cambiar el color de fondo.
      * **void setEditable (boolean b):** este método permite activar o desactivar la opción de editar el texto.
      * **void setText(String t):** este método permite cambiar el texto del componente

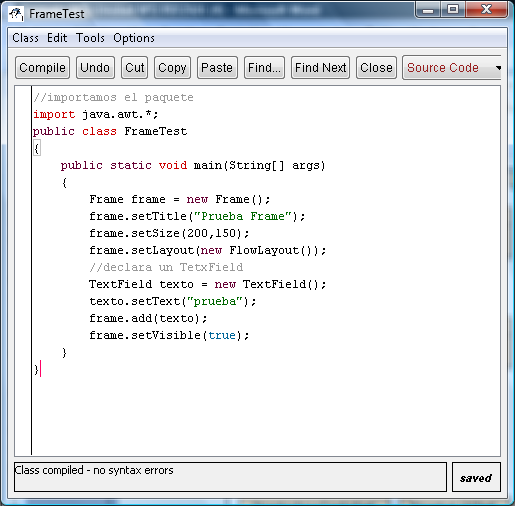
De la clase TextComponent heredan dos clases: TextField y TextArea.

**Clase TextField**

Esta clase hereda de TextComponent y representa un cuadro de texto donde se puede ingresar información en una línea.

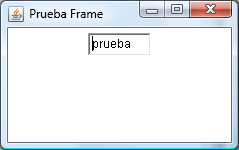
* + Constructor:
    - * **TextField() :** este constructor crea un cuadro de texto vacío
      * **TextField (int columnas):** este constructor crea un cuadro de texto vacío, con un tamaño indicado por columnas.
      * **TextField (String texto):** este constructor crea un cuadro de texto con el texto especificado en el parámetro de entrada
      * **TextField (String texto, int columnas):** este constructor crea un cuadro de texto con el texto especificado en el parámetro de entrada y el ancho especificado por columnas.
  + Utilidades: hereda todos los métodos de la clase TextComponent, además agrega nuevas utilidades tales como:
    - * **void addActionListener (ActionListener listener):** este método permite agregar un objeto escuchante de acción al cuadro de texto.
      * **boolean echoCharIsSet ():** verifica si el texto será encriptado, es decir aparece todo el texto representado con un carácter.
      * **char getEchoChar ():** este método devuelve el carácter con que se encriptará el texto.
      * **void setEchoChar (char c):** este método permite cambiar el carácter con que se encriptará el texto.
      * **void setText (String texto):** este método permite cambiar el texto que se despliega en el cuadro de texto.

Ejemplo de uso de un TextField en un contenedor:



En este ejemplo se crea un TextField con el contendido “prueba” y se agrega al contenedor.

El resultado de ejecutar el código anterior es el siguiente:



**Clase TextArea**

Esta clase hereda de TextComponent y representa un área donde se puede ingresar y manipular texto, en múltiples líneas.

* + Constructor:
    - * **TextArea ():** este constructor crea una nueva área de texto vacía con los dos Scrollbar (SCROLLBAR\_BOTH), vertical y horizontal.
      * **TextArea (int filas, int columnas**): este constructor crea una nueva área de texto vacía con los dos Scrollbar (SCROLLBAR\_BOTH), vertical y horizontal, pero el tamaño lo define filas y columnas en el parámetro de entrada.
      * **TextArea (String texto):** este constructor crea una nueva área de texto vacía con los dos Scrollbar (SCROLLBAR\_BOTH), vertical y horizontal, y se inicializa con el texto del parámetro de entrada.
      * **TextArea (String texto, int filas, int columnas):** este constructor crea una nueva área de texto vacía con los dos Scrollbar (SCROLLBAR\_BOTH), vertical y horizontal, y se inicializa con el texto del parámetro de entrada. Además el tamaño del área lo da filas y columnas declaradas en los parámetros.
      * **TextArea (String texto, int filas, int columnas, int scrollbars):** este constructor crea una nueva área de texto vacía con los dos Scrollbar (SCROLLBAR\_BOTH), vertical y horizontal, y se inicializa con el texto del parámetro de entrada. Además el tamaño del área lo da filas y columnas declaradas en los parámetros, y el tipo de barra de desplazamiento lo da el parámetro scrollbar, que puede ser:

SCROLLBARS\_BOTH,

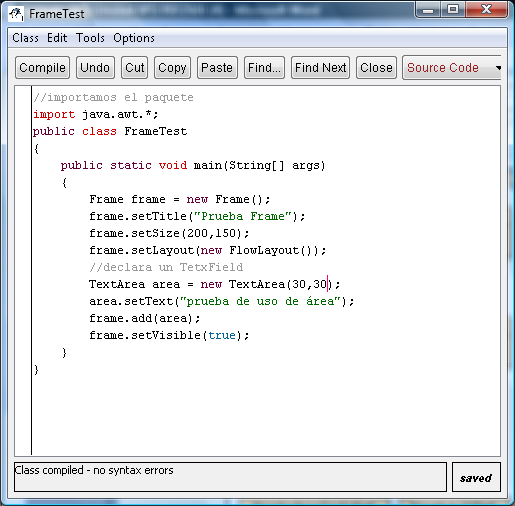
SCROLLBARS\_VERTICAL\_ONLY,

SCROLLBARS\_HORIZONTAL\_ONLY,

SCROLLBARS\_NONE.

* + Utilidades: esta clase utiliza todos los método heredados de la clase TextComponent, además agrega nuevas utilidades tales como:
    - **void append (String texto):** este método agrega el texto del parámetro a continuación del último texto ingresado en el área de texto.
    - **int getScrollbarVisibility ():** este método retorna un entero que identifica el tipio de barra que se está utilizando: vertical, horizontal ambas o ninguna.
    - **void insert (String texto, int posicion):** este método permite insertar en una posición el texto del parámetro, dentro del área de texto.
    - **void setColumns (int columnas):** este método permite cambiar el número de columnas que muestra el área de texto.
    - **void setRows (int filas):** este método permite cambiar el número de filas que muestra el área de texto.

Ejemplo de uso de TextArea en un contenedor:



Este ejemplo declara un TextArea, le agrega un texto y lo agrega al contenedor.

El resultado de ejecutar el código anterior es:

