**Cómo Utilizar las Clases TextArea y TextField**

Las clases [TextArea](http://java.sun.com/products/JDK/CurrentRelease/api/java.awt.TextArea.html) y [TextField](http://java.sun.com/products/JDK/CurrentRelease/api/java.awt.TextField.html) muestran texto seleccionable y, opcionalmente, permite al usuario editar ese texto. Se pueden crear subclases de TextArea y TextField para realizar varias tareas cómo comprobar los errores de la entrada. Como con cualquier componente, puede especificar los colores de fondo y de primer plano y la fuente utilizada en los campos y área de texto. Sin embargo no se puede cambiar su apariencia básica.

Estas dos clases, TextArea y TextField son subclases de [TextComponent](http://java.sun.com/products/JDK/CurrentRelease/api/java.awt.TextComponent.html). De esta clase heredan métodos que les permiten cambiar y obtener la selección actual, permitir o desactivar la edición, obtener el texto seleccionado actualmente (o todo el texto), y modificar el texto.

Abajo tiene un applet que primero muestra un TextField y luego un TextArea. El TextField es editable; y el TextArea no. Cuando el usuario pulsa la tecla Return en el Campo de Texto, su contenido se copia dentro del Área de Texto y luego lo selecciona en el Campo de Texto.

Aquí tienes el [programa](http://www.binarykode.com/bdescargas/Manuales%20y%20Documentos/JAVA/Interfaces%20de%20Usuario/Tutorial%20JAVA%20avanzado%20(I)/gui/clases/TextDemo.java). Aquí tienes el código que crea, inicializa, y maneja eventos del Campo y el Área de texto:

*//Donde se definan las variables de Ejemplar:*

TextField textField;

TextArea textArea;

public void init() {

textField = new TextField(20);

textArea = new TextArea(5, 20);

textArea.setEditable(false);

*...//Añade los dos componentes al Panel...*

}

public boolean action(Event evt, Object arg) {

String text = textField.getText();

textArea.appendText(text + "\n");

textField.selectAll();

return true;

}

La superclase TextComponente de TextArea y TextField suministra los métodos **getText()**, **setText()**, **setEditable()**, y **selectAll()** utilizados en el ejemplo anterior. También suministra los siguientes métodos: **getSelectedText()**, **isEditable()**, **getSelectionStart()**, y **getSelectionEnd()**. También proporciona un método **select()** que permite seleccionar el texto entre las posiciones de inicio y final que se especifiquen.

La clase TextField tiene cuatro constructores: **TextField()**, **TextField(int)**, **TextField(String)**, y **TextField(String, int)**. El argumento entero especifica el número de columnas del campo de texto. El argumento String especifica el texto mostrado inicialmente en el campo de texto. La clase TextField también suministra los siguientes métodos:

**int getColumns()**

Devuelve el número de columnas del campo de texto.

**setEchoChar()**

Activa el eco del caracter, es útil para los campos de Password.

**char getEchoChar()** y **boolean echoCharIsSet()**

Estos métodos le permite preguntar sobre el eco de caracteres.

Como la clase TextField, la clase TextArea tiene cuatro constructores: **TextArea()**, **TextArea(int, int)**, **TextArea(String)**, y **TextArea(String, int, int)**. Los argumentos enteros especifican el número de filas y columnas (respectivamente) del área de texto. El argumento String especifica el texto mostrada inicialmente en el área de texto.

La clase TextArea suministra el método **appendText()** utilizado en el ejemplo anterior. También suministra estos métodos:

**int getRows()**, **int getColumns()**

Devuelven el número de filas y columnas del área de texto.

**void insertText(String, int)**

Inserta el texto especificado en la posición indicada.

**void replaceText(String, int, int)**

Reemplaza el texto desde la posición inicial indicada (el primer entero) hasta la posición final indicada.

Como se puede imaginar por el nombre, JTextArea es un componente que crea un área de texto; este componente pertenece al paquete java.lang.Object y tiene algunas propiedades que debemos conocer:

La primera es que su tamaño es el justo para el texto que lleva dentro. Es decir, inicialmente, su tamaño es de una fila de letras de alto y cero pixeles de ancho, ya que no tiene texto. Según vamos escribiendo, va cambiando de tamaño. No es buena idea dejarlo así, ya que nos pueden pasar cosas raras. En el constructor del **JTextArea** podemos decirle cuántas filas queremos que tenga y cuántas columnas. En el ejemplo vamos a poner 10 filas y 50 columnas, pero también se puede agregar algún texto en nuestros JTextArea, como se verá a continuación:

*JTextArea jtx = new JTextArea()*; //sin parámetros   
*JTextArea jtx = new JTextArea(10,50)*; //número de filas y columnas   
*JTextArea jtx = new JTextArea( "Texto dentro de JTextArea" );*//cadena de texto

La segunda es que un **JTextArea** no tiene barras de scroll, así que si escribimos más de la cuenta, simplemente dejaremos de ver lo que escribimos. Para poner barras de scroll tenemos en Java el **JScrollPane**, un panel que admite dentro un componente. Para ingresar nuestro JTextArea dentro del JScrollPane debemos agregarlo como parámetro, y para que se activen los scroll debemos agregarle un comando específico para ello, este método se llama **setViewportView()** del **JScrollPane**y se implementa de la siguiente forma:

*JScrollPane scroll = new JScrollPane();*

*scroll.setViewportView(jtx);*

La tercera cosa que debemos saber es que el **JTextArea** por defecto no hace automáticamente los saltos de líneas. Es decir, si nosotros no hacemos de forma manual los saltos de línea, nuestro texto se escribirá en una línea de varios kilómetros. Para solucionar esto existen dos métodos: el primero corta las líneas de forma automática, pero no respeta las palabras, es decir las corta al momento de llegar al final de la línea sin importar si hay un espacio o no; el segundo método complementa al primero haciendo que las palabras de corten sólo cuando encuentra un espacio cerca. Los métodos son los siguientes:

// Para que haga el salto de línea en cualquier parte de la palabra:   
*jtx.setLineWrap(true);*

// Para que haga el salto de línea buscando espacios entre las palabras   
*jtx.setWrapStyleWord(true);*

Con todo esto, la forma de crear el **JTextArea** y añadirlo al **JFrame**será así:

*import javax.swing.\*;*

*import java.awt.\*;*

*public class VentanaEjemplo1{*

*public static void main(String[] args) {*

*JFrame ventana = new JFrame("Mi Ventana");*

*ventana.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);*

*JTextArea jtx = new JTextArea(10,50);*

*jtx.setLineWrap(true);*

*jtx.setWrapStyleWord(true);*

*JScrollPane scroll = new JScrollPane(jtx);*

*ventana.getContentPane().add(scroll, BorderLayout.CENTER);*

*ventana.add(scroll);*

*ventana.pack();//método utilizado para que se muestren los //elementos de la ventana*

*ventana.setVisible(true);*

*}*

*}*

Lo cuarto es que nuestros **JTextArea** pueden ser editados por el usuario o puede ser de sólo lectura, dependiendo de la propiedad setEditable(), el valor booleano false sólo da permiso de lectura:

*texto.setEditable(false);*

La quinta cosa importante que debemos mencionar es que podemos modificar es que podemos modificar muchas cosas a nuestro **JTextArea**, entre ellas están el tamaño, el tipo y la fuente de la letra como también podemos modificar el color de la letra y del fondo; la forma más simple de modificar las letras es crear un tipo de letra y luego entregarla como parámetro:

*Font font = new Font("Verdana", Font.ITALIC, 12);*

*jtx.setFont(font);*

Cuando creamos el tipo de fuente primero debemos entregar el nombre de la fuente, luego el formato en que la queremos: 

Normal=PLAIN

*Cursiva*=ITALIC

**Negrita**=BOLD

Y finalmente el tamaño de la letra. Para cambiar el color de la letra debemos utilizar el método:

*jtx.setForeground(Color.white);*

Para cambiar el color de fondo de nuestro **JTextArea** debemos utilizar el método:

*jtx.setBackground(Color.black);*

En ambos casos disponemos de varios colores ya designados:

*1. BLACK*

*2. BLUE*

*3. CYAN*

*4. DARK\_GRAY*

*5. GRAY*

*6. GREEN*

*7. LIGHT\_GRAY*

*8. MAGENTA*

*9. ORANGE*

*10. PINK*

*11. RED*

*12. WHITE*

*13. YELLOW*

Si no te gusta alguno de estos colores, puedes crear el tuyo utilizando la escala de colores **RGB** (utilizando valores entre 0 y 255), para implementarlo en cualquiera de los dos casos anteriores se realiza de la siguiente forma:

jtx.setForeground(new Color(int R,int G,int B));

jtx.setBackground(new Color(int R,int G,int B));

Finalmente veamos un ejemplo donde se muestran la mayoría de los componentes:

*import java.awt.\*;*

*import javax.swing.JFrame;*

*import javax.swing.JScrollPane;*

*import javax.swing.JTextArea;*

*public class JTextAreaTest{*

*public static void main (String [] args) {*

*//Método para cambiar la decoración de la ventana en sí,*

*//si ponemos el valor en falso la ventana se verá de la forma predeterminada de Windows*

*JFrame.setDefaultLookAndFeelDecorated(true);*

*//creamos el JFrame principal donde agregaremos el resto de objetos*

*JFrame frame = new JFrame ("JTextArea Test");*

*frame.setLayout (new FlowLayout ());*

*frame.setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);*

*String text = "Un objeto JTextArea es una área multilínea para escribir texto. "*

*+ "Se puede cambiar el número de líneas que se mostrarán, "*

*+ "y también el numero de columnas. Puedes cambiar las fuentes y tamaños de letras. "*

*+ "Y puedes agregar tu TextArea en un JScrollPane para poder mover las zonas de texto." ;*

*//creamos un area de texto normal no redimensionable*

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

*textAreal.setPreferredSize(new Dimension (10,100));*

*//creamos otra área de texto con un JScrollPane, sólo es agregado*

*JTextArea textArea2 = new JTextArea(text,20,10);*

*JScrollPane scroll = new JScrollPane(textArea2);*

*frame.getContentPane().add(scroll, BorderLayout.CENTER);*

*frame.add(scroll);*

*//El método setLineWrap lo que hace es ordenar las palabras para que no se salgan de los márgenes,*

*//si lo desactivamos(false), el texto se escribirá hacia el lado(en una línea) y no se vería todo si el //texto es muy largo*

*textAreal.setLineWrap (true);*

*textArea2.setLineWrap (true);*

*textAreal.setWrapStyleWord(true);*

*textArea2.setWrapStyleWord(true);*

*//agregamos los area de texto al frame principal,*

*//nota: el textArea2 está dentro de un JScrollPane, por eso éste es el que agregamos al frame*

*frame.add(textAreal);*

*frame.add(scroll);*

*//el metodo pack sirve para dejar todo dentro del frame*

*frame.pack();*

*//y finalmente el metodo setVisible es para que se visualice la ventana*

*frame.setVisible(true) ;*

*}*

*}*

***Ejercicios:***

-Crear una ventana simple que al ingresar texto lo organice de forma automática, realizando los saltos de líneas en cualquier parte de una frase, con fondo y tipo de letra a su elección.

-Crear una ventana con tres JTextArea: el primero que no realice saltos de líneas, con letra simple, el segundo debe tener Scrollbar, realizar los saltos de líneas en cualquier parte de la frase a escribir, con fondo amarillo y letra cursiva color negro, por ultimo el tercer cuadro debe tener Scrollbar con saltos de líneas buscando espacios, fondo color negro y letras blancas “en negrita”.