APRENDERAPROGRAMAR.COM

# INSTRUCCIÓN PEDIR EN JAVA CON JTEXTAREA Y MOSTRAR CON JLABEL. SETBOUNDS Y SETTEXT. EJEMPLOS. (CU00929C)

Sección: Cursos
Categoría: Lenguaje de programación Java nivel avanzado I

Fecha revisión: 2029

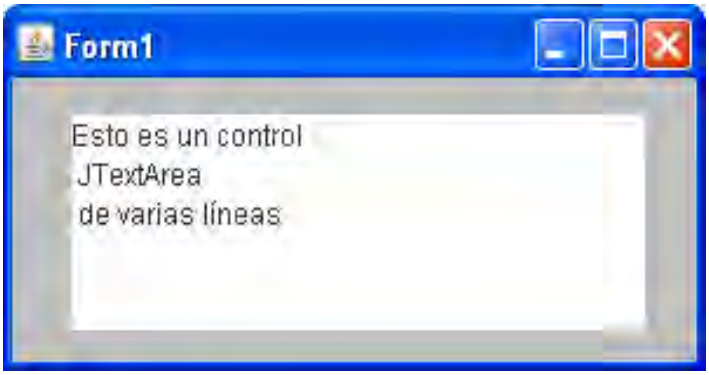
**Resumen:** Entrega nº29 curso "Lenguaje de programación Java Nivel Avanzado I".

Autor: Walter Sagástegui

## INSTRUCCIÓN PEDIR CON JAVA. JTextArea

JTextArea es un control para pedir varias líneas de texto, que a diferencia de JTextField, visto anteriormente, sólo nos permitía ingresar una línea de texto. Para mostrar su uso, recurriremos a la misma estructura visual básica que ya hemos utilizado anteriormente y que reproducimos en el siguiente código:


```
//(1) PAQUETE
import java.awt.*;
import javax.swing.*;

//(2) FORMULARIO
public class Programa extends JFrame {

  //(3) CONTROLES DEL FORMULARIO
  JPanel jpanel = (JPanel) this.getContentPane();
  JTextArea jtextarea = new JTextArea();

  //(4) CONSTRUCTOR DEL FORMULARIO
  public Programa() {

    //(5) PROPIEDADES DEL CONTENEDOR
    jpanel.setLayout(null);
    jpanel.setBackground(Color.lightGray);

    //(6) PROPIEDADES DE LOS CONTROLES
    jtextarea.setBounds(new Rectangle(25, 15, 250, 90));
    jtextarea.setText("Esto es un control \r\n JTextArea \r\n de varias líneas");
    jtextarea.setEditable(false);

    //(7) ADICION DE LOS CONTROLES AL CONTENEDOR
    jpanel.add(jtextarea, null);

    //(8) PROPIEDADES DEL FORMULARIO
    setSize(300,150);
    setTitle("Form1");
    setVisible(true);
  }
  //(9) METODOS DEL FORMULARIO
  public static void main(String arg[]) {
    new Programa();
  }
}
```




El código en todos sus detalles, fue explicado en la anterior entrega del curso. Sólo añadir, que en lugar del control JTextField estamos usando el control JtextArea. Al tratarse de un control para ingresar un texto de varias líneas, la forma de cambiar de línea es a través de los caracteres de escape **\r\n**, donde **\r** significa "ir al inicio de la línea" y \n significa "cambio de línea". Esto se puede observar en la instrucción siguiente:

**jtextarea.setText("Esto es un control \r\n JTextArea \r\n de varias líneas");**

El método **setText**, muestra el texto en el control JTextArea.

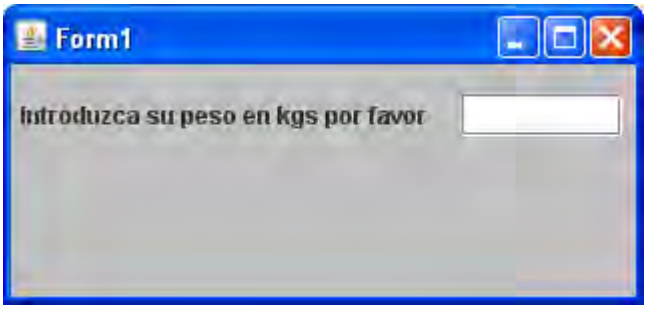
Indicar que si deseamos capturar el texto ingresado en un control JTextArea se puede usar el método **getText,** igual que para el control JTextField. Se utiliza de la siguiente manera:

**String s = jtextarea.getText().**
**String s = jtexfield.getText().**

**INSTRUCCIÓN MOSTRAR CON JAVA. Jlabels**

Un J*Label* sirve para insertar etiquetas que el usuario no podrá modificar en un formulario.

Vamos a crear un código de ejemplo donde se muestra en un Jlabel el mensaje "Introduzca su peso en kgs por favor" y donde el resultado que obtendremos es algo así:



Escribe y compila el siguiente código de ejemplo:

```
//(1) PAQUETE
import java.awt.*;
import javax.swing.*;

//(2) FORMULARIO
public class Programa extends JFrame {
  //(3) CONTROLES DEL FORMULARIO
  JPanel jpanel = (JPanel) this.getContentPane();
  JLabel jlabel = new JLabel();
  JTextField jtextfield = new JTextField();

  //(4) CONSTRUCTOR DEL FORMULARIO
  public Programa() {
    //(5) PROPIEDADES DEL CONTENEDOR
    jpanel.setLayout(null);
    jpanel.setBackground(Color.lightGray);

    //(6) PROPIEDADES DE LOS CONTROLES
    jlabel.setBounds(new Rectangle(5, 15, 220, 21));
    jlabel.setText("Introduzca su peso en kgs por favor");
    jtextfield.setBounds(new Rectangle(225, 15, 80, 21));

    //(7) ADICION DE LOS CONTROLES AL CONTENEDOR
    jpanel.add(jlabel, null);
    jpanel.add(jtextfield, null);

    //(8) PROPIEDADES DEL FORMULARIO
    setSize(300,150);
    setTitle("Form1");
    setVisible(true);
  }
  //(9) METODOS DEL FORMULARIO
  public static void main(String arg[]) {
    new Programa();
  }
}
```

En cuanto a las propiedades de los JLabels, su manejo es básicamente igual que el de los JTextField. Mostraremos a continuación una serie de métodos propios de un Jlabel. Estos métodos se usan para modificar las propiedades de los JLabels. Algunas están plasmadas en el ejemplo mostrado arriba y otras no. Vamos a dar ejemplos para un uso adecuado:

**jlabel.setFont (new java.awt.Font("Dialog", 1, 14));** El método **setFont** se usa para cambiar la fuente de letra a emplear. Si prestamos atención, se le pasa el objeto **new Java.awt.Font("Dialog",1,14) ,** que representa a la fuente de letra. El valor **"Dialog"** que se pasa como argumento es el nombre de la fuente, que podemos cambiar a cualquier otro tipo de fuente como **"Verdana", "Arial"** etc. El valor **1** indica el estilo negrita para la fuente, podemos cambiarla a **2** (estilo normal) o **3** (estilo cursiva). El valor 14 se refiere al tamaño de la letra que podemos también cambiar a nuestro gusto para que sea más grande o más pequeña.

**jlabel.setText ("Introduzca aquí un entero positivo");** El método **setText** se usa para establecer el texto que va a mostrar la etiqueta JLabel. Como vemos, el texto se le pasa como argumento.

**jlabel.setBounds (new Rectangle(130, 149, 125, 20));** El método **setBounds** se usa para posicionar el control JLabel en una posición determinada dentro del formulario. Esta posición está indicada por el objeto **new Rectangle(130, 149, 125, 20),** que se le pasa como argumento, donde **130, 149** son respectivamente las coordenadas **x,y** dentro del formulario y 125 es el ancho y 20 la altura del control JLabel.

**jlabel.setForeground (Color.red);** El método **setForeground** se usa para cambiar el color de letra del texto usado en la etiqueta. El argumento **Color.red** indica el color del texto que en este caso es rojo, pero podemos cambiar por cualquier otro color, por ejemplo **Color.green** (verde), **Color.black** (negro), **Color.white** (blanco), etc.

**jlabel.setHorizontalAlignment (SwingConstants.RIGHT);** El método **setHorizontalAlignment** se usa para alinear el texto dentro del control **Jlabel.** El argumento **SwingConstants.RIGHT** indica que el texto se debe alinear a la derecha, **SwingConstants.LEFT** sería para alinear a la izquierda y **SwingConstants.CENTER** al centro.

**jlabel.setBorder (BorderFactory.createLineBorder(Color.red, 1));** El método **setBorder** se usa para poner un borde a la etiqueta **JLabel.** Las propiedades del borde están indicadas por el objeto **BorderFactory.createLineBorder(Color.red, 1),** que se le pasa como argumento. **Color.red** indica el color rojo del borde y **1** es el grosor del borde. Mientras mayor sea el número más grueso será el borde.

Prueba a crear un Label y establece diferentes mensajes y propiedades para el mismo.

Para mostrar un mensaje compuesto de distintas partes en un JLabel, hemos de usar el símbolo de concatenación **+**. Por ejemplo:

```
jlabel.setText(msg1 + msg2);
```

El símbolo en cuestión, hace que se muestre el contenido derivado de las variables *msg1* y *msg2*.

Otra forma relevante de mostrar datos es a través de la ventana de Dialogo:

JOptionPane.showMessageDialog(ventana,"Mensaje","Titulo",JOptionPane.INFORMATION_MESSAGE);

Cuando se invoca la función JOptionPane.showMessageDialog, se despliega una ventana independiente donde se muestra un mensaje. A su vez, podemos configurar JOptionPane para que aparezcan distintos botones. Por ejemplo, si el mensaje es: "¿Quiere continuar?" los botones pueden ser "Sí", "No", o "Aceptar" y "Cancelar". Existen múltiples posibilidades como permitir que el usuario escriba o seleccione un valor de un combo, pero no vamos a entrar en ello ya que lo consideramos contenidos avanzados.

**Próxima entrega: CU00930C**

**Acceso al curso completo en aprenderaprogramar.com** -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=58&Itemid=180