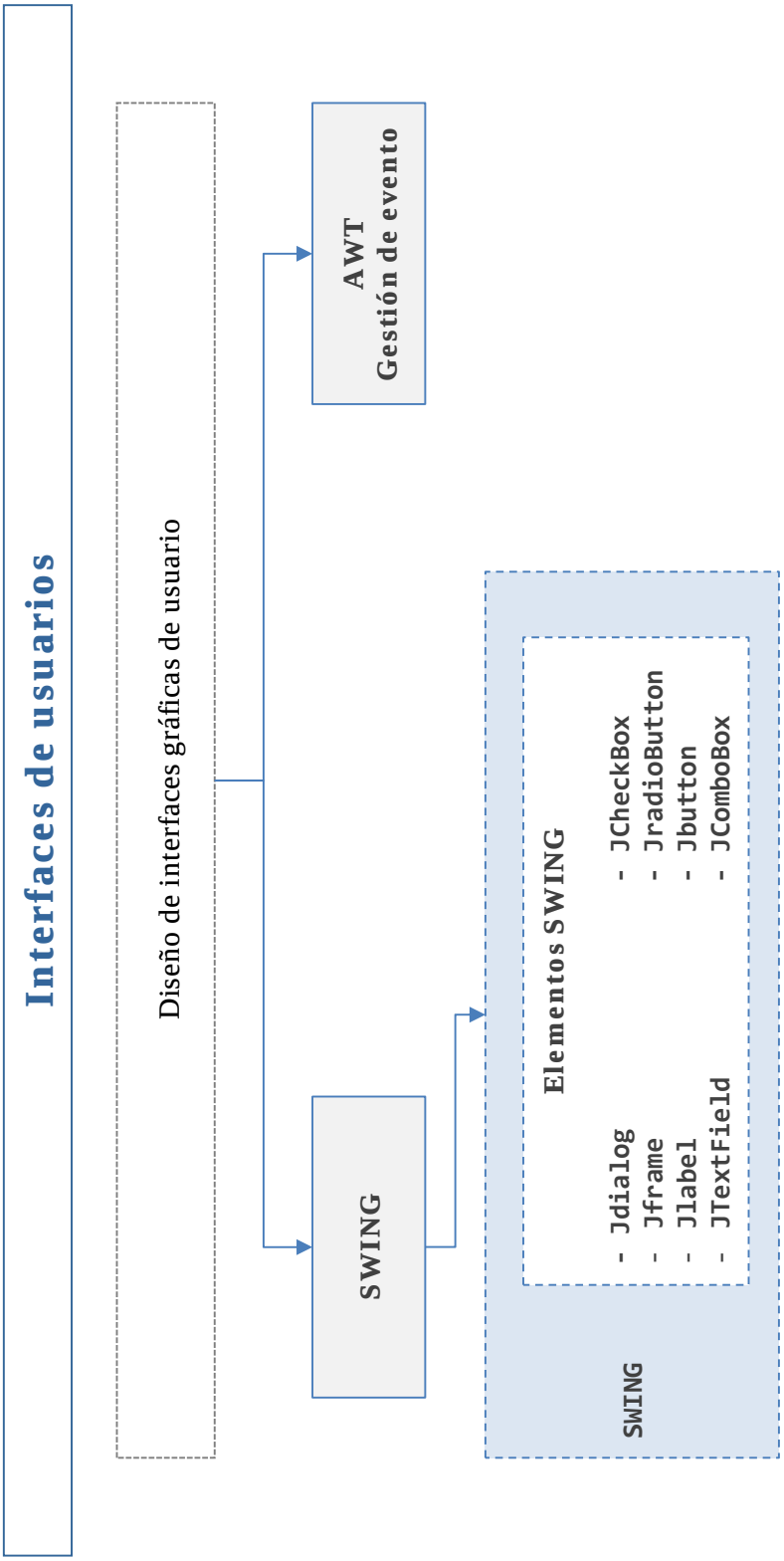
Interfaces de usuarios

TEMA 5

Esquema



Ideas clave

5.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee el capítulo 5, «Interfaz gráfica (GUI)» **(páginas 176-191)** del siguiente libro, disponible en el aula virtual en virtud del artículo 32.4 de la Ley de Propiedad Intelectual:

Sznajdleder, P. A. (2013). *Java a Fondo: estudio del lenguaje y desarrollo de aplicaciones* (2ª ed.). Buenos Aires: Alfaomega Grupo Editor.

Para comprobar si has comprendido los conceptos puedes realizar el test de autoevaluación del tema.

En este tema analizaremos qué son los eventos y cómo los podemos utilizar para crear interfaces de usuario. Veremos:

- Introducción a los eventos.
- Introducción a las interfaces de usuario.
- SWING.

5.2. Qué son los eventos

Evento

Suceso producido en un sistema cuando **el usuario realiza una acción**, por ejemplo, pulsar un botón o hacer clic sobre una opción de menú.

Para atender los eventos, los objetos fuentes (`Source`) se encargan de detectar los eventos y notificarlos a los objetos oyentes (`Listener`). Los objetos oyentes atienden los eventos que se registraron.

Los `listener` u objetos oyentes implementan una determinada interfaz `Listener`.

5.3. Desarrollo de interfaces de usuario

Interfaz Gráfica de Usuario (GUI)

Es un **conjunto de componentes** (ventanas, menús, botones, etc.), que permiten al usuario interactuar con la aplicación.

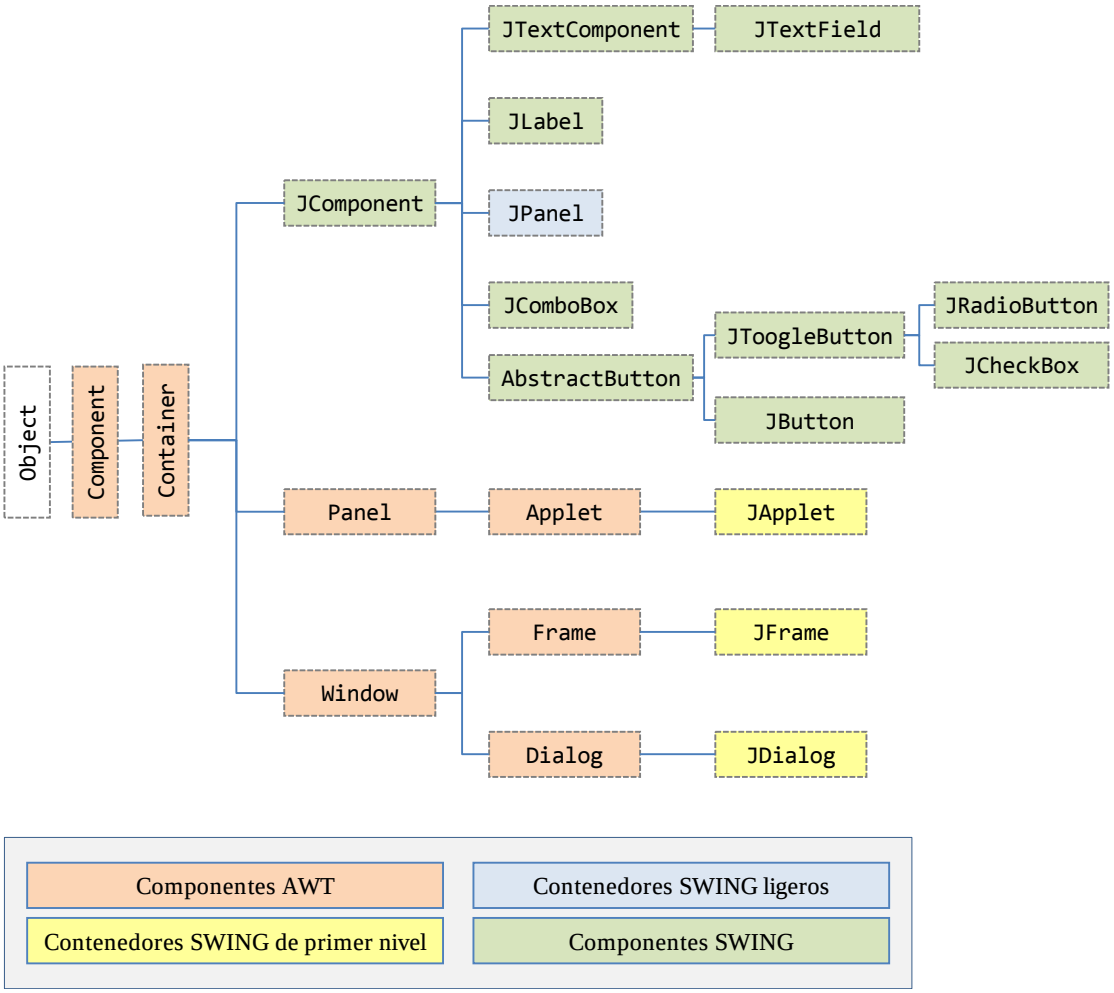
A menudo, cuando vamos a crear una interfaz gráfica de usuario, debemos decidir entre SWING o AWT.

A continuación se muestran las principales diferencias:

- AWT (Abstract Window Toolkit):
  - Herramientas originales de Java para la creación de interfaces de usuario.
  - Soportado por JDK 1.0 y 1.1.
  - Utiliza código nativo de la plataforma en la que se ejecuta.
  - No incluye componentes complejos.
  - Utiliza un modelo de manejo de eventos robusto.
  - Los componentes no se comportan igual en todas las plataformas.
- SWING:
  - Sus componentes están construidos sobre AWT.
  - Soportado por JDK 1.2
  - No utiliza código nativo.
  - Amplio conjunto de componentes
  - Los componentes se comportan igual en todas las plataformas.

5.4. SWING

En el siguiente diagrama se puede ver parte de la jerarquía de clases de SWING.



En una aplicación SWING debe existir, al menos, un **contenedor de primer nivel**, como por ejemplo un `JFrame`, que constituye la ventana principal. El resto de componentes cuelgan de este contenedor.

Ejemplo:

```
class Ventana extends JFrame{

   final int ALTO = 200;
   final int ANCHO = 300;
   Ventana(String titulo){
        setTitle(titulo);
        setSize(ANCHO,ALTO);

   }
}
```

`JFrame`

Es el **contenedor (marco)** que se utiliza para **situar en él todos los componentes**.

Constructores:

Algunos de sus constructores son:

- `JFrame()`: crea un marco sin título.
- `JFrame(string título)`: crea un marco con el título indicado.

Métodos:

Algunos de los métodos más relevantes son:

- `setTitle (string título)`: establece el título de la ventana.
- `setSize(int alto, int ancho)`: establece el ancho y alto de la ventana.
- `show()`: muestra el marco y sus componentes.
- `hide()`: oculta el marco y sus componentes.
- `setLocation(int x, int y)`: sitúa el marco en la posición x-y indicada.

Ejemplo:

```
import javax.swing.*;
class MiVentana extends JFrame {
       public MiVentana() {
       setTitle("Mi Ventana");
       setSize(300,200);
       }
}
public class ejemploFrame{
       public static void main(String[] args) {
       JFrame f = new MiVentana();
       f.show();
       }
}
```


JPanel

Para colocar el resto de componentes en un marco (*frame*), no se recomienda añadir directamente los componentes sobre la ventana, sino utilizar componentes menos pesados como los paneles.

La clase `JFrame` posee un **panel de contenido**, sobre el que situar los componentes, el cual se obtiene mediante el método `getContentPane()`

Ejemplo:

```
class MiVentana extends JFrame {
      public MiVentana() {
            setTitle("Mi Ventana");
            setSize(300,200);
            Container c=getContentPane();
            JLabel etiq = new JLabel ("Mi Etiqueta");
            c.add(etiq);
      }
}
```

Constructores:

Algunos de sus constructores son:

- `JPanel()`: crea un panel
- `JPanel(boolean b)`: crea un panel con doble buffer si el segundo argumento es `true`.
- `JPanel(Layout gestor)`: crea un panel con el gestor de posicionamiento indicado. Por defecto será `FlowLayout`.

Métodos:

Algunos de los métodos más relevantes son:

- `add(Component c)`: añade el componente al panel.
- `add(Component c, int pos)`: añade el componente al panel en la posición indicada dentro del panel.

- `add(Component c, String nombre)`: añade el componente al panel con el nombre indicado.
- `remove(Component c)`: elimina el componente del panel.
- `removeAll()`: elimina todos los componentes del panel.

`JDialog`

Se utiliza para crear **cuadros de diálogo**. Únicamente tienen un botón para cerrarlo, pero no para minimizarlo.

Los cuadros de diálogo se asocian a una ventana, por ejemplo a un `JFrame`, o a otro `JDialog`.

Pueden configurarse como modales, lo cual implica que los componentes de la ventana asociada estén bloqueados mientras el `JDialog` esté visible.

Constructores:

Algunos de los constructores de `JDialog` son:

- `JDialog()`: construye un objeto no modal sin marco asociado y sin título.
- `JDialog(JFrame frame, string titulo)` : construye un `JDialog` no modal, asociado al `JFrame` indicado en el primer argumento y con el título indicado en el segundo argumento.
- `JDialog(JDialog dialog, string titulo)`: construye un `JDialog` no modal, asociado al `JDialog` indicado en el primer argumento y con el título indicado en el segundo argumento.

- `JDialog(JFrame frame, boolean modal)`: construye `JDialog` asociado al `JFrame` indicado como primer argumento y con el comportamiento modal indicado en el segundo argumento.

Métodos:

Algunos de los métodos más relevantes son:

- `setModal(boolean modal)`: establece si es modal o no
- `setVisible(boolean v)`: establece si es visible o no.

`JLabel`

Permite **crear etiquetas** en las que se sitúan textos o imágenes.

Constructores:

Algunos de los constructores de `JLabel` son:

- `JLabel()`: crea una etiqueta sin texto.
- `JLabel (String texto)`: crea una etiqueta con el texto indicado.
- `JLabel (Icon imagen)`: crea una etiqueta con la imagen indicada.
- `JLabel(String texto, int alineación)`: crea una etiqueta con el texto indicado y con la alineación especificada en el segundo parámetro (RIGTH, LEFT, CENTER).
- `JLabel (String texto, Icon imagen, int alineación)`: crea una etiqueta con el texto, imagen y alineación indicados.

Métodos:

Algunos de los métodos más relevantes son:

- `setText (String texto)`: establece el texto de la etiqueta.
- `getText()`: obtiene el texto de la etiqueta.
- `setIcon(Icon imagen)`: establece la imagen de la etiqueta.
- `getIcon()`: obtiene la imagen de la etiqueta.
- `setVerticalAlignment(int alineación)`: establece la alineación vertical de la etiqueta.

- `getVerticalAlignment()`: obtiene la alineación vertical de la etiqueta.
- `setHorizontalAlignment(int alineación)`: establece la alineación horizontal de la etiqueta.
- `getHorizontalAlignment()`: obtiene la alineación horizontal de la etiqueta.

`JTextField`

Permite crear **campos de texto modificables** por el usuario.

Constructores:

Algunos de los constructores de `JTextField` son:

- `JTextField()`: crea un campo de texto vacío.
- `JTextField (String texto)`: crea un campo de texto con el texto especificado.
- `JTextField (int columnas)`: crea un campo de texto vacío con el número de columnas indicado.
- `JTextField (String texto, int columnas)`: crea un campo de texto con el texto y el número de columnas especificado.

Métodos:

Algunos de los métodos más relevantes son:

- `setText (String texto)`: establece el texto del campo de texto.
- `getText ()`: obtiene el texto del campo de texto.
- `setColumns(int columnas)`: establece el número de columnas del campo de texto.
- `getColumns()`: obtiene el número de columnas del campo de texto.
- `setEditable(boolean edit)`: establece si el campo de texto es editable.
- `isEditable()`: obtiene si el campo de texto es editable.

`JCheckBox`

Se utiliza para crear **casillas seleccionables**, permitiendo seleccionar más de una dentro de un grupo.

Constructores:

Algunos de los constructores de `JCheckBox` son:

- `JCheckBox()`: crea una casilla sin texto.
- `JCheckBox (String texto)`: crea una casilla con el texto especificado
- `JCheckBox (String texto, boolean seleccionado)`: crea una casilla con el texto especificado. Si seleccionado es true, la casilla aparece seleccionada.
- `JCheckBox (Icon imagen)`: crea una casilla con la imagen indicada en el argumento.
- `JCheckBox(Icon imagen, boolean seleccionado)`: crea una casilla con la imagen indicada. Si seleccionado es true, la casilla aparece seleccionada.

Métodos:

Algunos de los métodos más relevantes son:

- `isSelected ()`: obtiene si la casilla está seleccionada.
- `setSelected(boolean seleccionado)`: establece si la casilla está seleccionada.

`JRadioButton`

Se utiliza para crear **botones de radio**, permitiendo seleccionar solo uno dentro de un grupo.

Constructores:

Algunos de los constructores de `JRadioButton` son:

- `JRadioButton ()`: crea un botón sin texto.
- `JRadioButton (String texto)`: crea un botón con el texto especificado
- `JRadioButton (String texto, boolean seleccionado)`: crea un botón con el texto especificado. Si seleccionado es true, el botón aparece seleccionada.
- `JRadioButton (Icon imagen)`: crea un botón con la imagen indicada en el argumento.
- `JRadioButton (Icon imagen, boolean seleccionado)`: crea un botón con la imagen indicada. Si seleccionado es true, la casilla aparece seleccionada.

Métodos:

Algunos de los métodos más relevantes son:

- `isSelected ()`: obtiene si el botón está seleccionado.
- `setSelected(boolean seleccionado)`: establece si el botón está seleccionado.

`ButtonGroup`:

Para crear un grupo de botones, en el que solo uno puede estar seleccionado, se utiliza la clase `ButtonGroup`. Para añadir botones al grupo se utiliza el método `add`.

`JButton`

Se utiliza para **crear botones**.

Constructores:

Algunos de los constructores de `JButton` son:

- `JButton ()`: crea un botón sin texto.
- `JButton (String texto)`: crea un botón con el texto especificado.
- `JButton (Icon imagen)`: crea un botón con la imagen indicada en el argumento.
- `JButton (String texto, Icon imagen)`: crea un botón con el texto y la imagen indicados.

Métodos:

Algunos de los métodos más relevantes son:

- `setText(String texto)`: establece el texto del botón.
- `getText()`: obtiene el texto del botón.
- `setIcon(Icon imagen)`: establece la imagen del botón.
- `getIcon()`: obtiene la imagen del botón.
- `isSelected ()`: obtiene si el botón está seleccionado.
- `setSelected(boolean seleccionado)`: establece si el botón está seleccionado.

JComboBox

Permite la creación de **cuadros combinados**. Básicamente constan de una lista desplegable.

Constructores:

Algunos de los constructores de `JComboBox` son:

- `JComboBox ()`: crea un `ComboBox` vacío.
- `JComboBox (ComboBoxModel model)`: crea un `ComboBox` con la lista indicada.
- `JComboBox (Vector v)`: crea un `ComboBox` con la lista indicada.
- `JComboBox (Object[] l)`: crea un botón con el texto y la imagen indicados.

Métodos:

Algunos de los métodos más relevantes son:

- `setEdictable (boolean edit)`: si `edit` es `true`, permite al usuario introducir los valores.
- `addItem(Object o)`: añade un elemento en la lista.
- `insertItemAt(Object o, int pos)`: inserta un elemento en la lista en la posición indicada.
- `getItemAt(int n)`: obtiene el elemento n de la lista.
- `getItemCount()`: obtiene el número de elementos de la lista.
- `getSelectedItem()`: obtiene el elemento seleccionado.
- `removeItem(Object o)`: elimina el elemento indicado de la lista.
- `removeItemAt(int pos)`: elimina el elemento de la posición indicada.
- `removeAllItems()`: elimina todos los elementos de la lista.

Lo + recomendado

No dejes de leer...

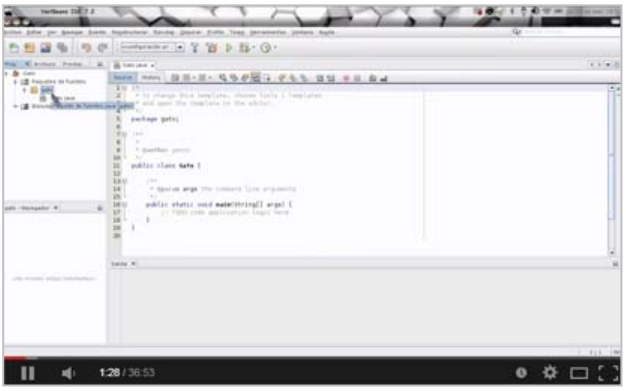
Oracle

En los siguientes enlaces accederás a la documentación oficial sobre componentes de SWING.

- `JFrame`: https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html
- `JPanel`: https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html
- `JDialog`: https://docs.oracle.com/javase/7/docs/api/javax/swing/JDialog.html
- `JLabel`: https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html
- `JTextField`: https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html
- `JCheckBox`: https://docs.oracle.com/javase/7/docs/api/javax/swing/JCheckBox.html
- `JRadioButton`: https://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html
- `JButton`: https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html
- `JComboBox`: https://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html

No dejes de ver...

Tres en raya



El siguiente vídeo explica cómo hacer un tres en raya en Java.

Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
http://www.youtube.com/watch?v=tS7RsVAOZsI

+ Información

A fondo

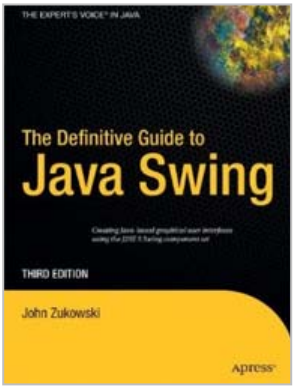
SWING de Oracle

Esta lección ofrece la información básica para utilizar los componentes SWING. Además, describe cada componente SWING.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://docs.oracle.com/javase/tutorial/uiswing/components/index.html

SWING

Zukowski, J. (2005). *The Definitive Guide to Java Swing*. Nueva York: Springer.



Libro en el que se hace un seguimiento de Swing en profundidad.

Accede a una parte del libro desde el aula virtual o a través de la siguiente dirección web:
http://books.google.es/books?id=YPjZNlEgAMcC&printsec=frontcover

Bibliografía

Fischer, P. (2005). *Introduction to Graphical User Interfaces with Java Swing*. Addison Wesley.

Test

1. Un evento es:

A. Un suceso que ocurre en tiempo de compilación.

B. Un suceso que ocurre cuando hay un error en el código.

C. Un suceso que ocurre en el sistema.

D. Un suceso que ocurre cuando se realiza la instanciación de objetos.

2. La diferencia entre AWT y SWING es:

A. SWING utiliza código nativo y AWT no.

B. No hay diferencia, son lo mismo.

C. AWT utiliza código nativo y SWING no.

D. AWT contiene SWING.

3. Las interfaces gráficas suelen tener:

A. Un contenedor de primer nivel, componentes de interfaz gráfica y gestión de eventos.

B. Un contenedor de primer nivel y gestión de eventos.

C. Con un contenedor de primer nivel se puede crear una interfaz completa.

D. Solo es necesaria la gestión de eventos.

4. Para crear una interfaz gráfica de usuario sobre un `Frame` hay que:

A. Heredar de la clase `JFrame`.

B. Instanciar la clase `JFrame`.

C. Solo hay que crear un objeto normal y luego asociarle el comportamiento.

D. No existen los `Frame` en Java.

5. Para indicar el título de un marco debemos:

A. Indicarlo en el constructor.

B. Indicarlo en el constructor o establecerlo invocando el método `setTitle(String Titulo)`.

C. Se indica invocando el método `setTile(String Titulo)`.

D. No se puede poner título.

6. `JDialog` se utiliza para:

A. Crear cuadros de diálogo.

B. Para crear un contendor de primer nivel.

C. No existe en Java.

D. Es un componente para organizar los elementos, pero no se visualiza.

7. Para obtener el elemento seleccionado de un `ComboBox` utilizamos el método:

A. `getSelectedItem()`.

B. `getItemAt(int pos)`.

C. `addItem(Object o)`.

D. `getItemCount()`.

8. Para pedir una cadena de texto al usuario utilizamos:

A. `JLabel`.

B. `JTextField`.

C. `JButton`.

D. `JCheckBox`.

9. Las casillas de verificación

A. Se crean con `JRadioButton` y pueden tener más de una opción seleccionada.

B. Se crean con `JCheckBox` y pueden tener más de una opción seleccionada.

C. Se crean con `JCheckBox` y solo pueden tener una opción seleccionada.

D. Se crean con `JRadioButton` y solo pueden tener una opción seleccionada.

10. Podemos poner un icono en:

A. Botones y botones de radio.

B. Botones, botones de radio y casillas de verificación.

C. Botones, botones de radio y etiquetas.

D. Botones, botones de radio, casillas de verificación y etiquetas.