GUI, Interfaz Gráfica de Usuario

Programación

EscuelaArteGranada

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN

- GUI: Graphic User Interface.

- Cuando se hace referencia a una interfaz gráfica de usuario o GUI, nos estamos refiriendo a una interfaz de ventanas. Las interfaces gráficas se caracterizan por una serie de componentes como son los botones, cajas de texto, etiquetas, barras de desplazamiento etc.

- En java, existen principalmente 2 librerías para desarrollar aplicaciones con interfaz gráfica, la **AWT** y la **SWING**

1. INTRODUCCIÓN

- **AWT** (Abstarct Windows toolkit). Es una biblioteca para desarrollar aplicaciones que se parezcan mucho al kit nativo del equipo en el que se instale la aplicación (mac, windos, linux…). La librería necesaria para utilizar estos métodos es la java.awt.*;

- **Swing** La ventaja que aporta swing frente a AWT es que se trata de una librería no nativa, lo que la hace más portable, aunque las aplicaciones desarrolladas no se integren tan bien. Los elementos de esta biblioteca son más potentes que los de AWT y se caracterizan por comenzar por J. La biblioteca necesaria para trabajar con ellos es javax.swing.*;

2. CREACIÓN DE VENTANAS

- Para crear una ventana con la biblioteca SWING se utiliza la clase **JFrame** de la cual tenemos que heredar para crear nuestra ventana**:**

```
public class NombreClase extends JFrame{
```

- Para configurar el aspecto y propiedades de la ventana se utilizan los siguientes métodos que se deben ejecutar desde el constructor:

- **pack();** //ajusta el tamaño de la ventana según los elementos que contenga
- **setSize(int, int);** //define el tamaño de la ventana (ancho,alto)
- **setVisible(boolean);** //hace visible u oculta una ventana
- **setLocation(int,int);** //posición de la ventana desde la esquina sup. Izq.
- **setLocationRelativeTo(component);** //posición de la ventana respecto a otra (null centrada en la pantalla)

2. CREACIÓN DE VENTANAS

- Otro método muy importante sobre el comportamiento de una ventana es el método:

setDefaultCloseOperation(modo);

- Este método establece la acción a realizar cuando se cierra la ventana, según el modo indicado:

- **JFrame.Hide_On_Close** //esconde la ventana
- **JFrame.Exit_On_Close** //cierra TODA la aplicación
- **JFrame.Dispose_On_Close** //cierra la ventana
- **JFrame.Do_Nothing_On_Close** //no hace nada

3. AÑADIR COMPONENTES A LA VENTANA

- Dentro de una ventana existen distintos tipos de componentes, todos ellos son clases y su utilización no difiere a la de cualquier otro objeto.

- Una vez creados e inicializados, para añadirlos a una ventana basta con invocar al siguiente método de la ventana deseada:

ventana. add(componente);

3. AÑADIR COMPONENTES A LA VENTANA

- Además de añadir los componentes a una ventana, hay que indicar como se van a ubicar dentro de la misma.

- Para esto existen el método **setLayout(obj)** que establece como se organizan los componentes dentro de una ventana según el objeto que reciba. Para utilizar estos métodos puede ser necesario importar java.awt.*;

- **setLayout(new FlowLayout())** //organiza los objetos de izquierda a derecha. Suele ser el valor por defecto.
- **setLayout(new GridLayout(filas, columnas))** //organiza los componentes en las filas y columnas indicadas.

4. COMPONENTES

- Los componentes más comunes proporcionados por SWING son:

Componente	Descripción	Constructor
JButton	Botón	JButton(“nombre”)
JLabel	Etiqueta	JLabel(“texto”)
JTextField	Línea de Texto	JTextField(“valor inicial”,tam)
JTextArea	Cuadro de Texto	JTextArea(“valorinicial”,tam)
JCheckBox	Casilla verificación	JCheckBox(“nombre”,booleano)
JRadioButton	Botón de opción	JRadioButon(“nombre”,booleano)
JComboBox	Lista Desplegable	JComboBox(String[] opciones)

4. COMPONENTES

- **JButton:** Es la clase que permite crear botones en una interfaz gráfica.
- Los métodos más comunes para crear y modificar un botón son:

JButton Botón = new JButton(“Nombre”);//crea el botón.

setText(“Nombre”); //establece la etiqueta del botón.

getText();//devuelve la etiqueta del botón.

getName();//devuelve el nombre del botón.

4. COMPONENTES

- JLabel: Es la clase que permite crear etiquetas de texto en una ventana.

- Los métodos más comunes para crear y modificar una etiqueta son:

  JLabel etiqueta = new JLabel("Texto");//crea la etiqueta.

  setText("Nombre"); //establece la etiqueta de la etiqueta.

  getText();//devuelve la etiqueta de la etiqueta.

  getName();//devuelve el nombre de la etiqueta.

4. COMPONENTES

- JTextField: Es la clase que permite cuadros de texto en los que puede escribir el usuario. Son campos de una sola línea

- Los métodos más comunes para crear y modificar cuadro de texto son:

JTextField cuadro = new JTextField("valor inicial");//crea el campo.

setText("Texto"); //establece el texto del campo.

getText();//devuelve el texto del campo.
getName();//devuelve el nombre del campo.
setEditable(boolean); Establece si el campo es editable o no por el usuario
isEditable(); //Devuelve si el campo es editable o no por el usuario.

4. COMPONENTES

- JTextArea: Es la clase que permite cuadros de texto en los que puede escribir el usuario. Son campos multilínea

- Los métodos más comunes para crear y modificar área de texto son

  JTextArea area = new JTextArea("valor inicial");//crea el campo.

  setText("Texto"); //establece el texto del campo.

  getText();//devuelve el texto del campo.

  append();//añade texto al campo.

  getName();//devuelve el nombre del campo.

  setEditable(boolean); Establece si el campo es editable o no por el usuario

  isEditable(); //Devuelve si el campo es editable o no por el usuario.

4. COMPONENTES

- JCheckBox: Son casillas de selección y permiten seleccionar una, varias o ninguna opción.
- Los métodos más comunes para crear y modificar un checkBox son

JCheckBox casilla = new JCheckBox("etiqueta",booleano);//crea la casilla.

setText("Texto"); //establece el texto la casilla.

getText();//devuelve el texto de la casilla.

getName();//devuelve el nombre de la casilla.

setSelected(boolean); Establece si la casilla esta marcado o no.

isSelected(); //Devuelve si la casilla está seleccionada o no.

4. COMPONENTES

- JRadioButton: Son similares a las casillas de selección pero su selección suele ser excluyente.
- Los métodos más comunes para crear y modificar un radioButton son

JRadioButton op = new JRadioButton (“valor”,booleano);//crea la casilla.

setText(“Texto”); //establece el texto la casilla.

getText();//devuelve el texto de la casilla.

getName();//devuelve el nombre de la casilla.

setSelected(boolean); Establece si la casilla esta marcado o no.

isSelected(); //Devuelve si la casilla está seleccionada o no.

¿pero como se hace para que sean excluyentes?

4. COMPONENTES

- Para que un grupo de JRadioButton sean excluyentes hay que agruparlos dentro de un mismo grupo:

ButtonGroup grupo = new ButtonGroup ();//crea el grupo de botones.

grupo.add(nombre); añade el botón al grupo.

Todos los elementos incluidos dentro de un ButtonGroup son excluyentes

4. COMPONENTES

- JComboBox: Lista desplegable sobre la cual seleccionar una opción. Los métodos más comunes para crear y modificar un radioButton son

JComboBox lista = new JComboBox (string[] opciones);//crea la lista.

El array de string contiene las opciones del ComboBox

getSelectedItem(); //devuelve el item seleccionado (null si no hay ninguno)
getSelectedInex(); //devuelve la posicion del item seleccionado
getName();//devuelve el nombre del ComboBox

4. COMPONENTES

- Cuando en una misma ventana tenemos varios componentes, es util agruparlos en grupos (paneles) de forma que estos se puedan manejar como uno sólo.
- Para esto se utiliza la clase JPanel:

JPanel nombre = new JPanel();

Los Jpanel poseen distintos métodos para establecer su borde, grosor nombre. Por ejemplo:

nombre.setBorder(BorderFactory.createTitledBorder("Título"));

nombre.setSize(ancho,alto);

- Para añadir un componente a un JPanel se utiliza el método:

nombre_panel.add(componente);

5. DIÁLOGOS

- Además de las ventanas principales de la interfaz, también suele ser interesante mostrar pequeñas ventanas de diálogo que muestran un mensaje o piden algún tipo de valor o confirmación.

- Esto se realiza mediante los objetos **JOptionPane**. Concretamente veremos estos 3 tipos:

  ShowMessageDialog: Muestra una ventana

  ShowConfirmDialog: Muestra un mensaje y pide confirmación (si, no)

  ShowInputDialog: Pide que se introduzca un valor.

- Estos diálogos siempre van a depender de una ventana padre.

5. DIÁLOGOS

- **ShowMessageDialog:** Muestra una pequeña ventana con un mensaje y un botón de aceptar que al pulsarlo cierra la ventana.

JOptionPane.showMessageDialog(JFrame, String) ;

Jframe: Nombre de la ventana de la que depende.

String: Mensaje a mostrar.

5. DIÁLOGOS

- **ShowConfirmDialog:** Muestra una pequeña ventana con un mensaje y los botones de si, no, cancelar. Devolviendo un entero que indica el botón que se ha pulsado.

  int resp=JOptionPane.showConfirmDialog(JFrame, String) ;

  Jframe: Nombre de la ventana de la que depende.

  String: Mensaje a mostrar.

  El valor devuelto es 0 si pulsa si, 1 si pulsa no y 2 si pulsa cancelar.

5. DIÁLOGOS

- **ShowInputDialog:** Muestra una pequeña ventana con un mensaje un cuadro de texto para responder. Devuelve un string con el valor introducido.

  String resp=JOptionPane.showInputDialog(JFrame, String) ;

  Jframe: Nombre de la ventana de la que depende.

  String: Mensaje a mostrar.

  Devuelve un string con el valor introducido por el usuario.

6. ACCIONES Y EVENTOS

- En una aplicación, cuando un usuario interactúa con la interfaz gráfica suceden cosas. Pulsar un botón, cerrar una ventana, seleccionar una casilla…

- Por tanto, el funcionamiento de una interfaz gráfica, generalmente es quedarse a la espera, (escucha) hasta que ocurran estas interacciones del usuario para darles respuesta.

- Para realizar esto existen una serie de manejadores/controladores de eventos (listeners) que son los encargados de ejecutar dichas respuestas.

6. ACCIONES Y EVENTOS

- Los eventos más comunes son:
  - **ActionListener:** Controlan la pulsación de botones, selección de casillas, pulsar intro en un campo de texto…
  - **MouseListener:** Detectan movimientos y clics del ratón.
  - **WindowListener:** Acciones con una ventana como cerrarla redimensionarla, minimizarla…
  - Y muchos más: **FocusListener, KeyListener, ItemListener…**

En este tema trabajaremos principalmente con **ActionListener.**

6. ACCIONES Y EVENTOS

- Cuales son los pasos para crear y gestionar eventos:
  - Crear el componente que va a reaccionar al evento (boton, check, cuadro de texto…)
  - Añadir el Listener que se va a activar cuando ocurra la acción deseada
  - Programar un método o métodos que gestione la respuesta al evento
  - Es necesario añadir la biblioteca java.awt.event.*;

6. ACCIONES Y EVENTOS

- Añadir un Listener:

```
//creamos el componente
JButton Botón = new JButton("Nombre");

//añadimos el listener
Botón.addActionListener(new ActionListener){
        public void actionPerformed(ActionEvent evt) {
                métodoRespuesta(evt);
    }
}
```

La respuesta se puede implementar directamente aquí o en métodoRespuesta.

Si se unifican varios Listener, podemos saber quién lo originó mediante el método del evento getSource()

6. ACCIONES Y EVENTOS

```
public class Ventana extends JFrame{

   private JLabel etiqueta;
   private JTextField campo;
   private JButton enviar;
   …
   public Ventana(){
    super("Prueba");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   this.setSize(500, 500);
   etiqueta = new JLabel("como te llamas");
   campo= new JTextField(30);
   boton=new JButton(“Enviar”);
   this.setLayout(new FlowLayout());
   this.add(eti);
   this.add(campo);
   this.setVisible(true);
   }

}
```

6. ACCIONES Y EVENTOS

- Acciones del interfaz de usuario:

```
boton.addActionListener(newActionListener({
        public void actionPerformed(ActionEvent e){
                JOptionPane.showMessageDialog(vent, "hola"+campo.getText());
        }
 });


campo.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e){
                JOptionPane.showMessageDialog(vent, "hola"+campo.getText());
  }
 });
```

7. BIBLIOGRAFÍA

- García de Jalón, j.: "*Aprende Java como si estuvieras en primero*". Editorial TECNUN. 2000

- Holzner, S.: "*La biblia de JAVA 2*". Editorial Anaya Multimedia 2000.

- Schildt, H.: "*Java, manual de referencia*". Editorial McGraw-Hill 2009.

7. BIBLIOGRAFÍA

- Oracle and/or its affiliates: "*Java ™ Platform, Standard Edition 7 API Specification*". Última visita: Febrero 2014.
  http://download.oracle.com/javase/6/docs/api/

- Oracle and/or its affiliates: "*Java SE Documentation at a Glance*". Última visita: Febrero 2014.
  http://download.oracle.com/javase/7/docs/api/

- Programación en castellano: "Java". Última visita: Febrero 2014.
  http://www.programacion.com/java

7. BIBLIOGRAFÍA

- Wikipedia, la enciclopedia libre. Última visita: Febrero 2014. http://es.wikipedia.org/

- López, J.C.: "*Curso de JAVA* Última visita: Febrero 2014. http://www.cursodejava.com.mx

- V.V.A.A. .: "*Guía de iniciación al lenguaje Java*". Última visita: Febrero 2014. http://zarza.usal.es/%7Efgarcia/doc/tuto2/Index.htm