INTERFAZ GRÁFICA DE USUARIO (GUI)

Hasta ahora todos los programas interactuaban a través de consola, esto lo hace más tedioso y aburrido, y que estamos acostumbrados a trabajar con la interfaz gráfica.

Java nos da dos clases para trabajar con interfaces graficas.

* Awt: es la clase más antigua y depende del sistema operativo.
* Swing: salió más tarde, y no depende del sistema operativo.

Que hay que tener en cuenta para crear una GUI.

* Contenedores
* Componentes
* Distribuidor de componentes (Layouts…)
* Gestión de Eventos

CONTENEDORES:

Se utilizan para dibujar los componentes.

* JFrame🡪 La ventana principal
* JDialog🡪Ventana secundaria.
* JPanel🡪Panel

COMPONENTES:

Son los botones, etiquetas, tablas, etc. Estos no se añaden directamente al JFrame, si no al panel contenido en el JFrame o a uno creado por nosotros.

This.getContentPane()

* JLabel🡪etiqueta
* JButton🡪Boton
* JTextField🡪Linea de texto
* JTextArea🡪Area de texto
* JPassWordField🡪\*\*\*\*
* JCheckBox 🡪Casilla
* JRadioButton🡪Casilla de selección
* JComboBox🡪Desplegable
* JTable🡪tabla que nospuede interesar para hacer bases de datos
* JList🡪Lista de elementos

Cuando quereos anular el distribuidor por defecto que trae el panel empleamos.

panel.setLayout(null)

Generalmente se utiliza los layouts manager para distribuir los componentes.

Hay varios layouts

* FlowLayout 🡪Por defecto del panel. Distribuye los componentes en horizontal
* BoxLayout🡪Permite poner los componentes tanto en el eje x como en el eje y esto también se lo indicamos al constructor.
* BorderLayout🡪 Nos pone los componentes en los puntos cardinales N,S,E,W
* GridLayout🡪Para representar una tabla

Puede ser que en nuestro Frame queramos representar dos distribuciones diferentes, para esto haríamos varios paneles.

**EVENTOS:**

Hasta ahora en la GUI cuando pulsamos en un botón vemos que no pasa absolutamente nada.

Cuando el usuario genera una acción sobre un componente, esta acción se llama evento, Y la aplicación debe generar una respuesta ajustada a ese evento.

Para eso debemos añadir “Listeners” para informar al programa y dar una respuesta acertada

Los Listeners, son interfaces, para cada evento hay un Listener/interfaz determinada. Ej: mouseListener, ActionListener.

MouseListener

Metodo1();🡪 implements MouseListener{}

Metodo1{

loquesea}

Metodo2(){}vacio si no lo voy ausar.

En java, cuando tenemos una interfaz que tiene que implementar muchos métodos, se crea una clase llamada Adapter 🡪 MouseAdapter

Esta clase tendría: Metodo1,Metodo2,Metodo3,Metodo4,Metodo5 (en blanco)

Ahora si usamos MouseAdapter, en vez de implementar tenemos que extender y aquí se sobreescribiría.

Imaginemos que vamos a hacer un evento de ratón, en el cual solo vamos a usar un único método, que usamos de ellos dos, Listener o Adapter.

Imaginemos que más adelante vamos a tener que crear un método nuevo, pues esta con Listener no funcionaría ya que estamos usando métodos de más que están en la implementación.

Que pasos debemos seguir para trabajar con Eventos.

1. Implementar o extender la superclase correspondiente.

Public class VentanaEvento implements ActionListener{

1. Asociamos el Listener al componente

Button\_copiar.addActionListener()

1. Definir el método que se va ha ejecutar.

Public void actionPermormed(ActionEvent e){

Object boton = e.getSource() //sóbre que componente es el evento

If (botón ==bCopiar){

String result = jLabel.getText();

aTextcopia.setText(result);

aTextCopia.settext(ltextNome.getText());

}

}

¿Dónde definimos el ActionPerformer?

bCopiar.addActionListener (this) indica que esto se resolverá en la propia clase.

CALSE INTERNA:

Una clase interna es una clase definida dentro de otra clase. La clase exterior se llama clase externa o colectora.

Las clases internas permiten evitar, la proliferación de clases muy pequeñas. Que en muchos casos solo se utilizan dentro de una clase.

En las clases internas accedemos directamente a los atributos de las clases externas, incluidos los private.

CLASE ANONIMA:

Esta es una clase sin nombre, definida en la misma línea de código donde se crea el objeto de la clase, se utiliza con subclases o interfaces que tienen uno o dos métodos, y cada uno de ellos con muy poco código. (se emplea con interfaz gráfica).

bCopiar.addActionListener(new ActionListener(){aquí se crea la clase anónima

Public void actionPerformed( ActionEvent e){

areaTexto.setText(Text\_box.getText());

}

});

CARACTERISTICAS AVANZADAS EN LA PROGRAMACIÓN ORIENTADA A OBJETOS

HERENCIA Y POLIMORFISMO

La herencia es una de las características de la programación orientada a objetos. Tiene una seria de ventaja:

* Mejorar el diseño de nuestra aplicación.
* Reutilización de código
* Facilita la escalabilidad del programa

Sabemos también que cuando una clase hereda de otra, va a heredar todos aquellos atributos que no sean privados, y de los métodos hereda todos excepto el constructor.

Para crear un objeto con todo empleamos la palabra super();

Para acceder a la superclase hacemos super. Y con this. Accedemos a la clase actual

f.m1 1

f.m2 muestra el abuelo

f.m3 muestra el abuelo

n.m4 = 0

n.m1 =1

a.m1 = 0

a.m2 = “avo m2”