

Interfaces gráficas de usuario en Java con SWING

¿Qué es un GUI?

- Son interfaces gráficas en las que el usuario puede interactuar con el ratón o introducir datos a través del teclado en unas cajas de texto, o seleccionar opciones de radio buttons o checkbox, o pulsar botones para realizar distintas acciones.
- En JAVA se utiliza la librería SWING que contiene una serie de clases para crear este tipo de aplicaciones.

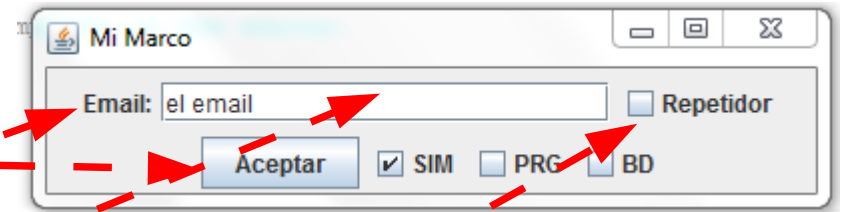
Creación de interfaces de usuario simples

- Usar componentes: botones, cajas de texto, etiquetas, y otras ...
- Agrupar componentes, poner componentes juntos
- Anidar componentes, poner componentes dentro de otras
- Abrir y cerrar ventanas

- En Java la creación de GUI's es multiplataforma.
- Se basa fundamentalmente en SWING
- AWT (Abstract Windowing Toolkit) : conjunto de clases que nos permiten crear una GUI e interactuar con el usuario:
 - Botones
 - Cajas de chequeo
 - Etiquetas
 - Campos de texto
 - Ventanas
 - Paneles
 - Cajas de diálogo
 - ...
 -

- En Java cada parte de una GUI, es una clase del paquete Swing:

- JButton
- JWindow
- JLabel
- JTextField
- JFrame
- JComboBox para los menús
- JCheckBox ..



SWING y AWT

- Para crear interfaces GUI, deberemos de importar los paquetes SWING y AWT, que contienen las clases a utilizar.
- A partir de ese momento utilizaremos esas clases para crear objetos, particularizar sus atributos y llamaremos a sus métodos, igual que hasta el momento hemos hecho con las aplicaciones de consola.

- Una GUI cuenta con dos tipos de objetos
 - Los componentes: botones, cajas de texto, etiquetas....
Elementos individuales que forman la GUI
 - Los contenedores, es el espacio donde se van a alojar los componentes. El primer paso será crear el contenedor para alojar los componentes. Los contenedores más habituales son:
 - Ventanas
 - Marcos
 - Cajas de diálogo

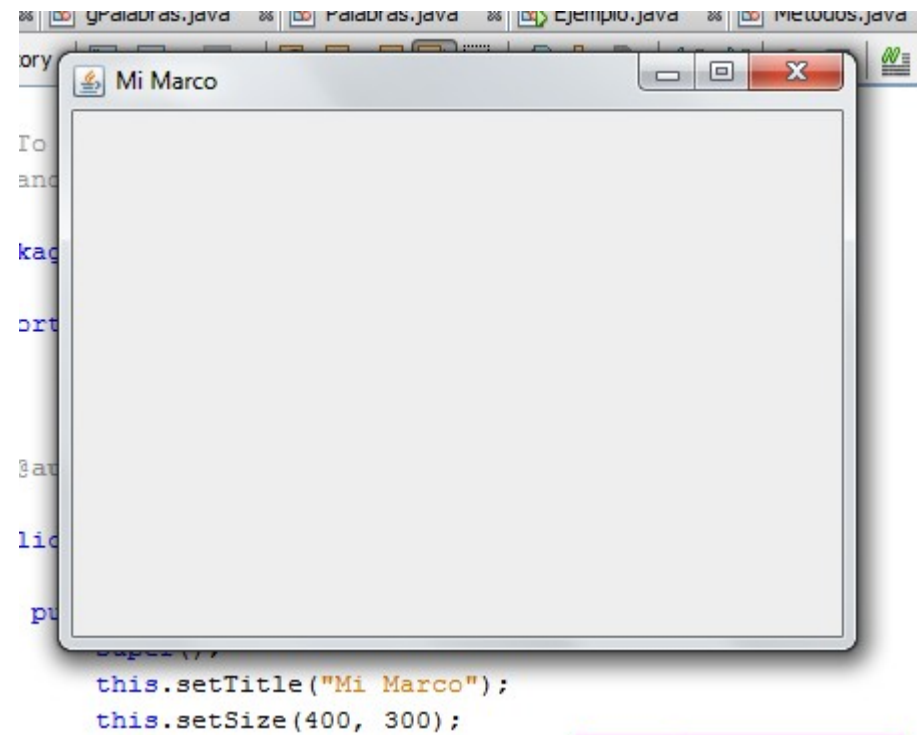
Los contenedores

- Las ventanas y los marcos son los principales contenedores que contendrán los componentes de la GUI. Son clases, que habrá que crear y configurar
- Las ventanas (JWindow) son los contenedores más simples que no tienen un título en la parte superior.
- Los marcos (JFrame) disponen de barra de título y botones, (minimizar, maximizar, cerrar)

Usar un Marco

- Para usar un marco en una aplicación JAVA hay que hacer una subclase de JFrame y configurarlo en el constructor para adaptarlo a nuestra aplicación:
 1. Llamar al constructor de la clase JFrame, llamando a “super()”
 2. Configurar el título del marco, lo podemos hacer de dos formas:
 - Poner el nombre al llamar al constructor padre: `super(“Mi marco”);`
 - Llamar al método `setTitle`
 3. Configurar el tamaño del marco, mediante el método `setSize`, pasándole el ancho y el alto o utilizando el método `pack()` después de añadir componentes dentro, y de adapta al tamaño de estos.
 4. Configurar el cierre de la ventana con el método `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
 - Hacerla visible con el método `setVisible(true);`

```
public class MiMarco extends JFrame {  
  
    public MiMarco() {  
        super();  
        this.setTitle("Mi Marco");  
        this.setSize(400, 300);  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        this.setVisible(true);  
    }  
}
```



Añadiendo componentes al contenedor

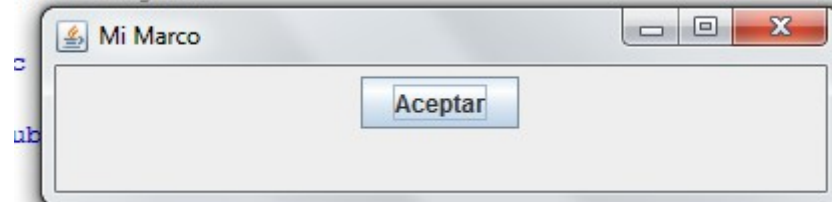
- El proceso será:
 - Crear y configurar la componente
 - Añadir la componente al contenedor con el método Add
 - Utilizar un gestor de diseño para disponer las componentes, por ejemplo FlowLayout es el más simple. Creamos el objeto
 - Y utilizamos el gestor de diseño con el método setLayout

El componente JButton

```
public class MiMarco extends JFrame {  
  
    public MiMarco() {  
        super();  
        this.setTitle("Mi Marco");  
        this.setSize(400, 100);  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        JButton miboton=new JButton("Aceptar");  
        this.add(miboton);  
        this.setVisible(true);  
        FlowLayout dis=new FlowLayout();  
        this.setLayout(dis);  
    }  
}
```

Layout Manager

author majesus



this.setTitle("Mi Marco");

Etiquetas y campos de texto

Nombre

Alineación

- JLabel email=
new JLabel("Email",JLabel.RIGHT);
- JTextField miemail=
new JTextField(20);

[valor inicial]

Nº caracteres

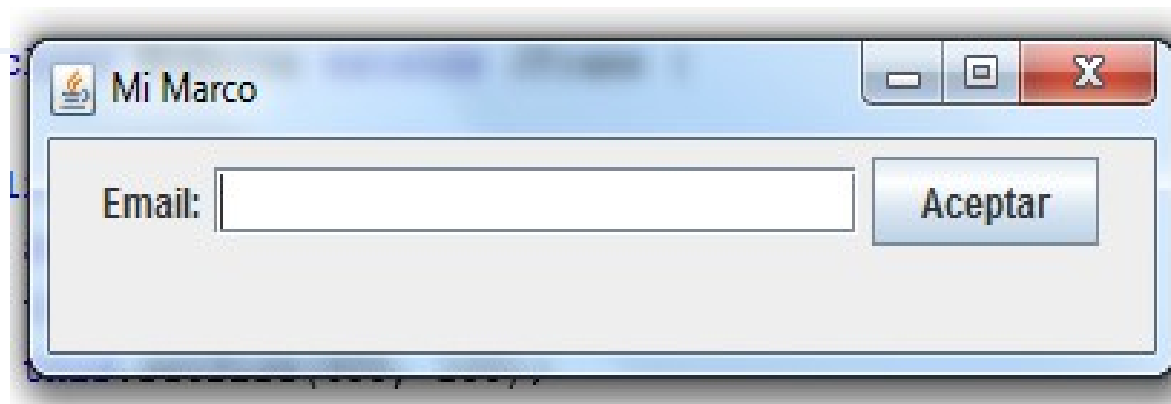
Para recuperar la información de una caja de texto utilizaremos el método `getText()`, y para asignarsela, el método `setText()`

```

public class MiMarco extends JFrame {

    public MiMarco() {
        super();
        this.setTitle("Mi Marco");
        this.setSize(400, 100);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel email=new JLabel("Email:",JLabel.RIGHT);
        JTextField miemail=new JTextField(20);
        this.add(email);
        this.add(miemail);
        JButton miboton=new JButton("Aceptar");
        this.add(miboton);
        this.setVisible(true);
        FlowLayout dis=new FlowLayout();
        this.setLayout(dis);
    }
}

```



JCheckBox: Casillas de verificación

- Estas componentes pueden ser presentadas solas o como parte de un grupo.
- `JCheckBox rep=new JCheckBox("Repetidor");`
- `this.add(rep);`



Diagram illustrating the visual representation of a JCheckBox component. It consists of a light blue rectangular box with a thin black border. Inside the box, the text "Texto del CheckBox" is centered. A thin black line extends from the top-right corner of the box towards the right, indicating its connection to the code in the list above.

Texto del CheckBox

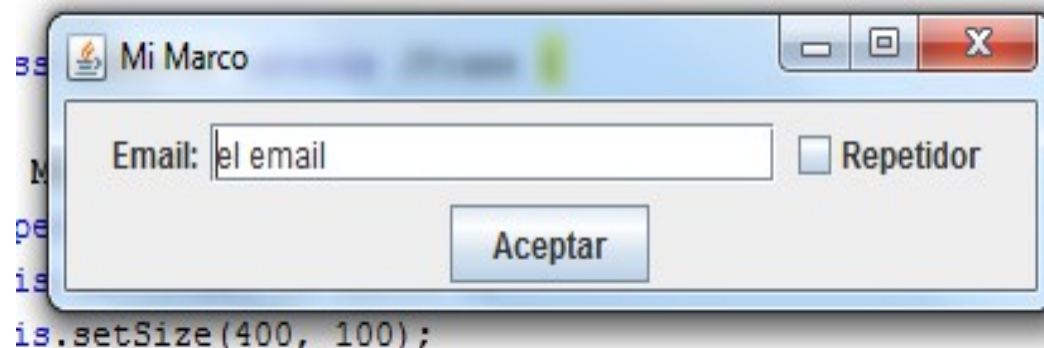
Si aparece como 2º argumento “true”, aparecerá la caja seleccionada por defecto

```

public class MiMarco extends JFrame {

    public MiMarco() {
        super();
        this.setTitle("Mi Marco");
        this.setSize(400, 100);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel email=new JLabel("Email:",JLabel.RIGHT);
        JTextField miemail=new JTextField(20);
        miemail.setText("el email");
        this.add(email);
        this.add(miemail);
        JCheckBox rep=new JCheckBox("Repetidor");
        this.add(rep);
        JButton miboton=new JButton("Aceptar");
        this.add(miboton);
        this.setVisible(true);
        FlowLayout dis=new FlowLayout();
        this.setLayout(dis);
    }
}

```



Agrupación de JCheckBox con ButtonGroup

- En caso de estar agrupadas, solo es seleccionable una de ellas, es decir, actúan como botones de radio.

```
JCheckBox m1=new JCheckBox("SIM",true);
JCheckBox m2=new JCheckBox("PRG",true);
JCheckBox m3=new JCheckBox("BD",true);
ButtonGroup grupo=new ButtonGroup();
grupo.add(m1);
grupo.add(m2);
grupo.add(m3);
this.add(m1);
this.add(m2);
this.add(m3);

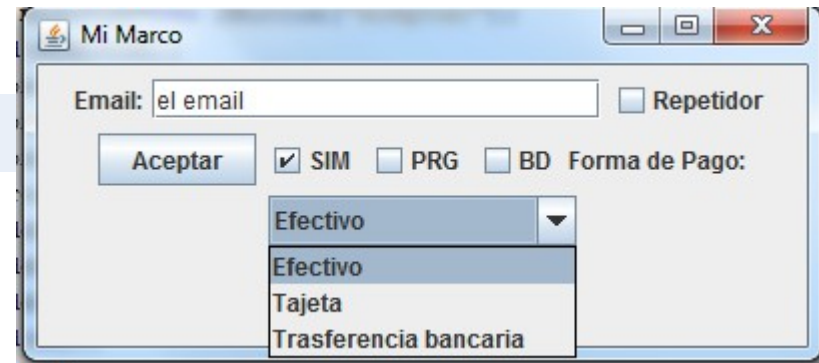
this.setVisible(true);
FlowLayout dis=new FlowLayout();
this.setLayout(dis);
```

JComboBox: Menús desplegables

- Con el método `addItem`, añadimos componentes, con el método `setEditable`, permitimos la inserción:

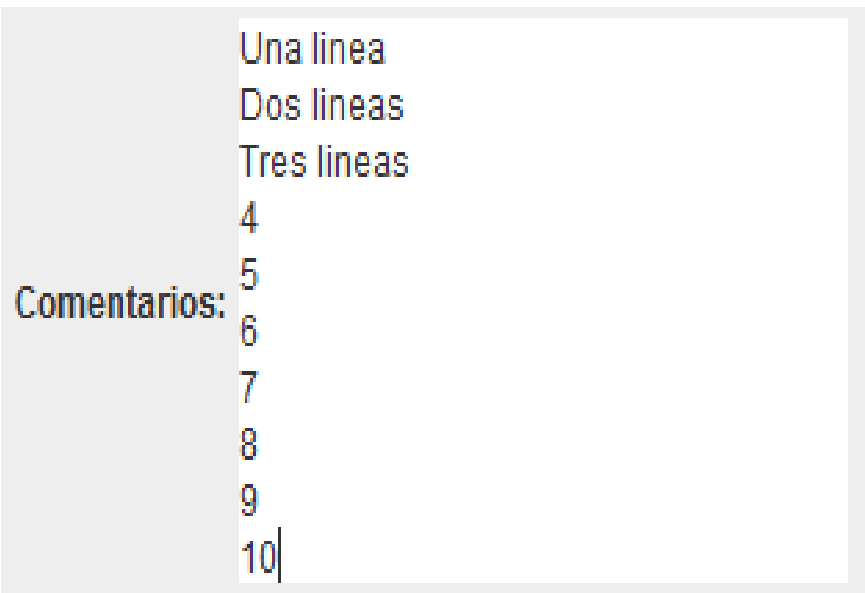
```
JLabel fp=new JLabel("Forma de Pago:",JLabel.RIGHT);
JComboBox menu=new JComboBox();
menu.addItem("Efectivo");
menu.addItem("Tajeta");
menu.addItem("Trasferencia bancaria");
this.add(fp);
this.add(menu);
FlowLayout dis=new FlowLayout();
this.setLayout(dis);
this.setVisible(true);
```

El método **`setEditable(true)`** permite introducir datos además de poderlos seleccionar



JTextArea: Areas de texto

- Son campos de texto de mayor capacidad que los JtextBox, en los que solo se podían introducir una línea.
- Se le pasan dos informaciones:
 - Alto en líneas
 - Ancho en nº de caracteres



```
JLabel comen = new JLabel("Comentarios:", JLabel.LEFT);  
JTextArea comentarios = new JTextArea(10, 20);  
this.add(comen);  
this.add(comentarios);
```

JPanel

- Es un tipo de contenedor que divide el área de una interface en secciones.
- La diferencia entre un JFrame y un JPanel es que los segundos son solo marcos que permiten organizar componentes y no tienen ni botones ni barra de título.
- Sirven para crear nuestras propias componentes que puedan ser agregadas a otras clases.

Creando un componente personalizado

- Vamos a crear un componente que muestra la fecha y la hora con un Jpanel.
- Posteriormente lo podemos utilizar en cualquier proyecto de la misma manera que hemos hecho con los componentes predefinidos.

MiReloj

- Necesitamos primero poder obtener la fecha y hora del sistema utilizando la clase Calendar, de la siguiente manera:

```
private String mes(int m) {
    String[] meses = {"", "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre",
        "Octubre", "Noviembre", "Diciembre"};
    return meses[m];
}

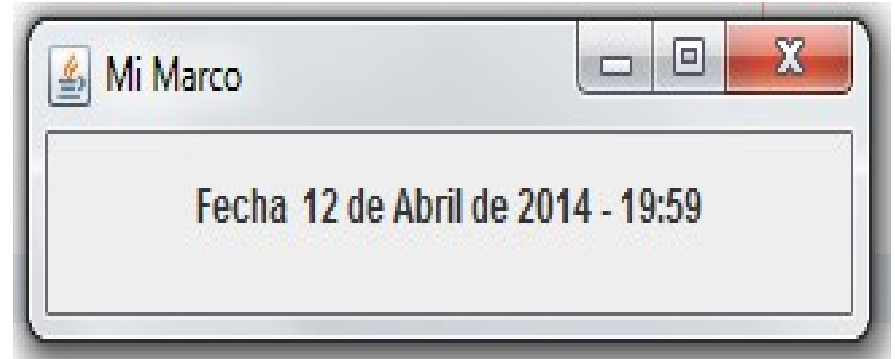
private String dameTiempo() {
    String tiempo = "";
    int dia, mes, año, hora, min;
    Calendar ahora = Calendar.getInstance();
    dia = ahora.get(Calendar.DAY_OF_MONTH);
    mes = ahora.get(Calendar.MONTH);
    año = ahora.get(Calendar.YEAR);
    hora = ahora.get(Calendar.HOUR_OF_DAY);
    min = ahora.get(Calendar.MINUTE);
    tiempo = dia + " de " + mes(mes) + " de " + año + " - " + hora + ":" + min;
    return tiempo;
}
```

MiReloj

```
public class mireloj extends JPanel {  
  
    public mireloj() {  
        super();  
        String tpo=dameTiempo();  
        JLabel etiqueta=new JLabel("Fecha");  
        JLabel fecha=new JLabel(tpo);  
        this.add(etiqueta);  
        this.add(fecha);  
    }  
  
    private String mes(int m) { ... }  
  
    private String dameTiempo() { ... }  
}
```


Utilizando el componente creado

```
public class MiMarco extends JFrame {  
  
    public MiMarco() {  
        super();  
        this.setTitle("Mi Marco");  
        this.setSize(600, 300);  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        mireloj mr=new mireloj();  
        this.add(mr);  
  
        FlowLayout dis = new FlowLayout();  
        this.setLayout(dis);  
        this.setVisible(true);  
    }  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
        MiMarco mm=new MiMarco();  
    }  
}
```

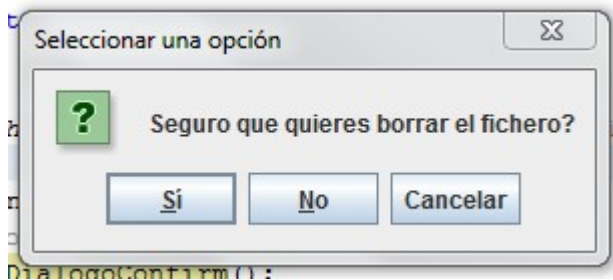


Cajas de diálogo: JOptionPane

- Son cajas predefinidas utilizadas para mostrar mensajes o pedir datos.
- Existen 4 tipos de cajas de diálogo standard:
 - `ConfirmDialog`. Hace una pregunta con botones para respuesta: si, no o cancelar
 - `InputDialog`: pide información al usuario a través de una caja de texto
 - `MessageDialog`. Muestra un mensaje en pantalla
 - `OptionDialog`. Engloba los otros 3 tipos anteriores

Métodos de JOptionPane

- Show: permite mostrar las cajas de diálogo

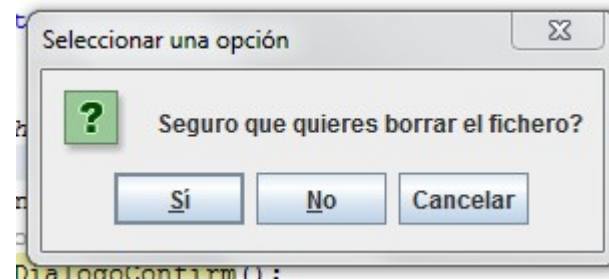


ConfirmDialog

- El método más sencillo de crearlas es con el método `showConfirmDialog`, al que le pasaremos 2 argumentos:
 - Componente: contenedor padre de la caja de diálogo o null (si utilizamos null no se abre en ningún componente en concreto)
 - Objeto: puede ser
 - un String: texto que aparecerá en la caja
 - Un componente
 - Un icono

- Este método nos puede devolver 3 posibles opciones:
 - YES_OPTION
 - NO_OPTION
 - CANCEL_OPTION





Un ejemplo:

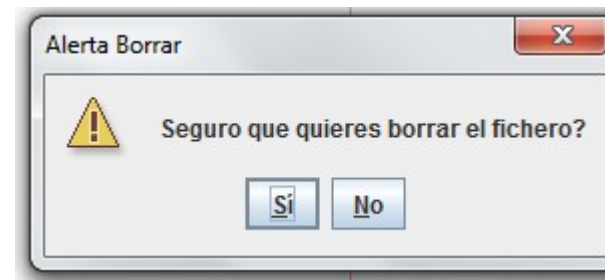


```
-  ~/  
public class DialogoConfirm extends JFrame {  
  
    public DialogoConfirm() {  
        int resp = JOptionPane.showConfirmDialog(null, "Seguro que quieres borrar el fichero?");  
    }  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
        DialogoConfirm mm = new DialogoConfirm();  
    }  
}
```

Mas opciones de creación

showConfirmDialog(componente,objeto,string,int,int)

- Los dos primeros tiene el mismo significado
- 3º, es una cadena que aparece en la barra de título de la caja de diálogo
- 4º, entero que especifica cuales botones queremos que aparezcan en la caja: dos constantes: YES_CANCEL_OPTION o YES_NO_CANCEL_OPTION
- 5º, entero que especifica el icono que queremos que aparezca en la caja de diálogo:
 - QUESTION_MESSAGE 
 - INFORMATION_MESSAGE 
 - WARNING_MESSAGE 
 - ERROR_MESSAGE 
 - PLAIN_MESSAGE sin icono

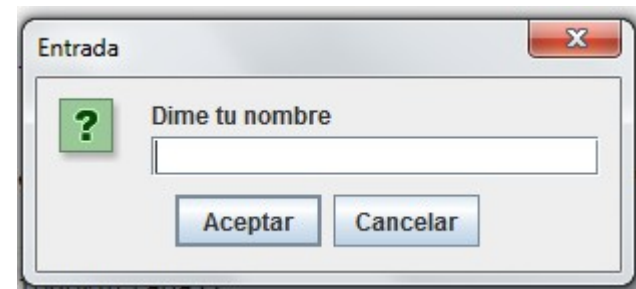


InputDialog

- Estas cajas de diálogo sirven para recoger datos.

`showInputDialog(componente , objeto);`

- Donde ambas tienen el mismo significado de antes:
 - Componente: contenedor padre
 - Objeto: string, componente o icono a mostrar antes de la caja de texto



```
public class DialogoEntrada extends JFrame{

    public DialogoEntrada() {
        String resp=JOptionPane.showInputDialog (null,"Dime tu nombre");
    }

    public static void main(String[] args) {
        DialogoEntrada de=new DialogoEntrada();
    }
}
```

Mas opciones de creación

showInputDialog(componente,objeto,string,int)

- Los dos primeros tiene el mismo significado
- 3º, es una cadena que aparece en la barra de título de la caja de diálogo
- 4º, entero que especifica el icono que queremos que aparezca en la caja de diálogo:

– QUESTION_MESSAGE



– INFORMATION_MESSAGE



– WARNING-MESSAGE



– ERROR_MESSAGE

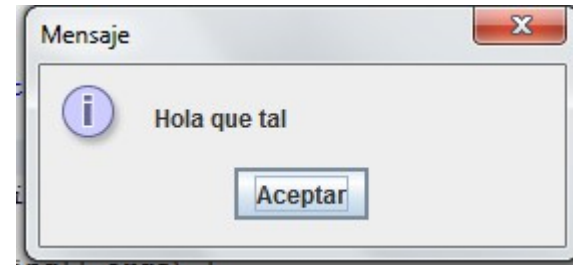


– PLAIN_MESSAGE sin icono

AlertDialog

- Caja de diálogo destinada a mostrar información
- La forma de crearla es similar a las anteriores
 - `showAlertDialog(componente,objeto)`
 - `showAlertDialog(componente,objeto,string,int)`
- En este caso no devuelve nada

Un ejemplo



```
public class DialogoMensaje extends JFrame{  
  
    public DialogoMensaje() {  
        JOptionPane.showMessageDialog(this, "Hola que tal");  
    }  
  
    public static void main(String[] args) {  
        DialogoMensaje dm=new DialogoMensaje();  
    }  
}
```