

El control botón

El control *Button*

El botón o control *Button* es uno de más simples y utilizados en un interfaz gráfico de usuario.

```
Button btnAceptar=new Button();
```

Para establecer el título del botón se llama a *setLabel*

```
btnAceptar.setLabel("Aceptar");
```

El control *Label*

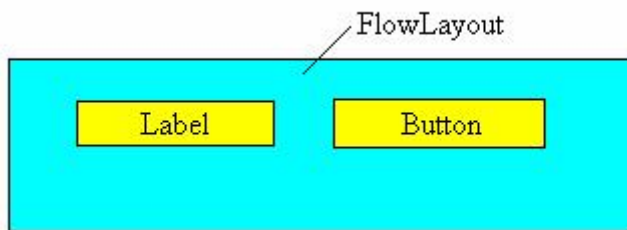
La etiqueta o control label sirve para mostrar un mensaje que habitualmente no cambia. Normalmente, acompaña a los controles de edición, para indicar al usuario el tipo de información que tiene que introducir. Para establecer el texto en la etiqueta se llama a *setText*

```
lMensaje.setText("Se ha pulsado el botón");
```

Propósito

Un control etiqueta muestra el texto "Pulsar el botón". Al pulsar en el botón Aceptar se cambia el texto que aparece en dicho control a "Se ha pulsado el botón"

Diseño



Crear un applet y situar sobre el applet en el modo diseño un control etiqueta (Label) y un botón (Button).

Cambiar sus propiedades en sus respectivas hojas de propiedades

Establecer [*FlowLayout*](#) como gestor de diseño del applet, separando horizontalmente los controles, cambiando la propiedad **hgap**.

Respuesta a las acciones del usuario

Para responder a la acción de pulsar un botón, hay que implementar el interface *ActionListener*. Dicho interface tiene una única función miembro denominada *actionPerformed*. La clase que implementa este interface obligatoriamente ha de definir esta función.

```
public interface ActionListener extends EventListener {
    public void actionPerformed(ActionEvent e);
}
```

Definimos una clase denominada *AccionBoton* que implementa el interface *ActionListener*. En *init* se asocia el control que es la fuente de los sucesos con un objeto (listener) de dicha clase que maneja los sucesos.

Se define una clase *AccionBoton* que implementa el interface *ActionListener* y define la función *actionPerformed*

```
class AccionBoton implements ActionListener{
    private Label label;
    public AccionBoton(Label label){
        this.label=label;
    }
    public void actionPerformed(ActionEvent ev){
        label.setText("Se ha pulsado el botón");
    }
}
```

Se asocia la fuente que produce los sucesos, el botón *btnAceptar*, con el objeto *accion* de la clase *AccionBoton* que maneja o que está interesado en dichos sucesos.

```
AccionBoton accion=new AccionBoton(lMensaje);
btnAceptar.addActionListener(accion);
```

La clase *AccionBoton* es independiente de la clase que describe el applet, *BotonApplet2*. La función miembro *actionPerformed* de la clase *AccionBoton* necesita tener acceso al control *lMensaje* para que muestre un texto en dicho control cuando se pulsa el botón. Para ello, ponemos como miembro dato de dicha clase un control etiqueta (*Label*) que inicializamos en el constructor al pasarle *lMensaje*.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class BotonApplet2 extends Applet{
    Label lMensaje = new Label();
    Button btnAceptar = new Button();
    FlowLayout flowLayout1 = new FlowLayout();

    public void init() {
        lMensaje.setText("Pulsar el botón          ");
        btnAceptar.setLabel("Aceptar");
        AccionBoton accion=new AccionBoton(lMensaje);
        btnAceptar.addActionListener(accion);
        flowLayout1.setHgap(25);
    }
}
```

```

        this.setLayout(flowLayout1);
        this.add(lMensaje, null);
        this.add(btnAceptar, null);

    }
}
//*****
class AccionBoton implements ActionListener{
    private Label label;
    public AccionBoton(Label label){
        this.label=label;
    }
    public void actionPerformed(ActionEvent ev){
        label.setText("Se ha pulsado el botón");
    }
}

```

Se mueve el código de la función respuesta a la clase que describe el applet

Estudiamos una pequeña modificación del ejemplo anterior, pero que es importante para el mantenimiento del código.

```

import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class BotonApplet3 extends Applet{
    Label lMensaje = new Label();
    Button btnAceptar = new Button();
    FlowLayout flowLayout1 = new FlowLayout();

    public void init() {
        lMensaje.setText("Pulsar el botón          ");
        btnAceptar.setLabel("Aceptar");
        AccionBoton accion=new AccionBoton(this);
        btnAceptar.addActionListener(accion);
        flowLayout1.setHgap(25);
        this.setLayout(flowLayout1);
        this.add(lMensaje, null);
        this.add(btnAceptar, null);
    }
    void funcionRespuesta(ActionEvent ev) {
        lMensaje.setText("Se ha pulsado el botón");
    }

}
//*****
class AccionBoton implements ActionListener{
    private BotonApplet3 applet;
    public AccionBoton(BotonApplet3 applet){
        this.applet=applet;
    }
    public void actionPerformed(ActionEvent ev){
        applet.funcionRespuesta (ev);
    }
}

```

Vemos que en la clase que describe el applet *BotonApplet3*, se define una función respuesta *funcionRespuesta*. Desde la función miembro *actionPerformed* de la clase *AccionBoton* se llama a dicha función. De este modo al ser *funcionRespuesta* miembro de la clase que describe el applet tiene acceso a todos sus miembros sean públicos o privados.

Otra ventaja que presenta esta aproximación es que en la clase que define al applet, *BotonApplet3*, tenemos por una parte el código de inicialización del applet, en *init*, y por otra el código de las funciones respuesta. El mantenimiento del código es más fácil con esta aproximación, ya que una sola clase controla la inicialización y las funciones respuesta a las acciones del usuario, en vez de estar en clases separadas.