

Grupo de controles

La información acerca del suceso (event)

En el estudio de la función respuesta a la acción de pulsar sobre un botón, o de hacer doble-clic sobre un elemento de una lista, el objeto *ev* de la clase *ActionEvent*, nos proporciona información acerca del suceso que se ha producido, nos dice qué control ha generado el suceso. Por ejemplo, podemos saber mediante la función miembro *getSource* si el suceso procede de la acción sobre un botón, de un control lista, o de un control de edición, etc.

En el código de la función respuesta *actionPerformed*, podemos saber si el control sobre el que se ha actuado es una instancia de la clase *Button* mediante el operador [instanceof](#)

```
Object control=ev.getSource();
if(control instanceof Button){
    System.out.println("Se ha pulsado un botón");
}
```

Si hay varios botones, podemos saber cual de ellos ha sido pulsado mediante [equals](#).

```
Object control=ev.getSource();
if(control.equals(btnAceptar){
    System.out.println("Se ha pulsado el botón Aceptar");
}
```

Mediante *getActionCommand* obtenemos el nombre (etiqueta) del botón. Podemos saber si se ha pulsado sobre un botón titulado "Rojo".

```
String nombre=ev.getActionCommand();
if(nombre.equals("Rojo")){
    System.out.println("Se ha pulsado el botón Rojo");
}
```

Como veremos más adelante, dependiendo del tipo de suceso, la información está encapsulada en distintas clases, que proporcionan la información que interesa al usuario. Así, la clase *AdjustmentEvent* encapsula toda la información referente a las acciones sobre la barra de desplazamiento. Del objeto *ev* podemos extraer mediante la función miembro *getValue*, el valor al que equivale la posición del dedo en la barra de desplazamiento.

```
void funcionRespuesta(AdjustmentEvent ev) {
    radio=ev.getValue();
    repaint();
}
```

La clase *ItemEvent* encapsula la información referente a las acciones sobre un control lista, o un control selección. La función miembro *getItem* obtiene de un objeto *ev* de dicha clase el elemento que ha sido seleccionado. *getItem* devuelve un objeto de la clase

base *Object*, que puede ser necesario promocionar (casting) a un objeto de la clase adecuada.

En el caso de un control selección (*Choice*) comparamos mediante *equals*, el objeto devuelto por *getItem* (el elemento seleccionado) con cada uno de los nombres de los elementos de dicho control.

```
void funcionRespuesta(ItemEvent ev) {  
    if((ev.getItem()).equals("Desayuno")) {  
        //...  
    }  
}
```

En el caso de un control lista (*List*) el objeto de la clase *Object* devuelto por *getItem*, es promocionado a *Integer*. El objeto de la clase *Integer* es convertido en un número entero (el índice del elemento seleccionado) mediante la función *intValue*.

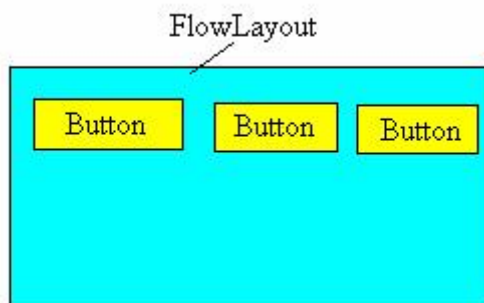
```
void funcionRespuesta(ItemEvent e) {  
    indice=((Integer)e.getItem()).intValue();  
    //...  
}
```

Respuesta a las acciones del usuario sobre un conjunto de tres botones

Propósito

Ya estudiamos la respuesta a la [acción de pulsar sobre un botón](#). Diseñamos un applet que contiene un conjunto de tres botones situados en la parte superior del applet. El nombre de los botones es el de los colores primarios: azul, verde y rojo. Al pulsar un botón se pinta un rectángulo de dicho color.

Diseño



Crear el applet, situar tres botones en la parte superior.

Cambiar sus propiedades **name** y **label** en sus respectivas hojas de propiedades

Establecer [FlowLayout](#), como gestor de diseño del applet. Cambiar la propiedad **hgap** para separar horizontalmente los botones.

Crear un array *colores* de objetos de la clase [Color](#) con los tres colores primeros: rojo, verde y azul.

```
final Color[] colores={Color.red, Color.green, Color.blue};
```

Redefinir la función *paint* para pintar un rectángulo

Respuesta a las acciones del usuario

Vamos a ver cómo se puede elaborar una respuesta única a las acciones sobre un conjunto de controles que responden de forma semejante.

Podemos pensar en un único objeto *accion* de una clase que implemente el interface *ActionListener* esté interesado en las acciones sobre los tres botones. Definimos una clase que implemente el interface *ActionListener* y defina la función *actionPerformed*.

```
class AccionBotones implements ActionListener{
    private BotonesApplet1 applet;
    public AccionBotones(BotonesApplet1 applet){
        this.applet=applet;
    }
    public void actionPerformed(ActionEvent ev){
        applet.respuestaBotones (ev);
    }
}
```

Creamos un objeto *accion* de la clase *AccionBotones* y le pasamos el applet. **this** inicializa el miembro dato *applet* de la clase *AccionBotones*. Desde dicho objeto llamamos a la función respuesta denominada *respuestaBotones*, en la definición de *actionPerformed*.

```
AccionBotones accion=new AccionBotones(this);
```

Asociamos mediante la función *addActionListener* cada uno de los botones con el objeto *accion* que registra las acciones sobre dichos botones.

```
btnRojo.addActionListener(accion);
btnVerde.addActionListener(accion);
btnAzul.addActionListener(accion);
```

Definimos la función respuesta *respuestaBotones*, la tarea a realizar, cuando se pulsa sobre alguno de los botones de la misma forma que en el ejemplo anterior.

El código completo de este ejemplo, es el siguiente

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

public class BotonesApplet1 extends Applet {
    Button btnRojo = new Button();
    Button btnVerde = new Button();
    Button btnAzul = new Button();
```

```

FlowLayout flowLayout1 = new FlowLayout();
final Color[] colores={Color.red, Color.green, Color.blue};
int indice;

public void init() {
    btnRojo.setLabel("Rojo");
    btnVerde.setLabel("Verde");
    btnAzul.setLabel("Azul");
    AccionBotones accion=new AccionBotones(this);
    btnRojo.addActionListener(accion);
    btnVerde.addActionListener(accion);
    btnAzul.addActionListener(accion);
    flowLayout1.setHgap(10);
    this.setLayout(flowLayout1);
    this.add(btnRojo, null);
    this.add(btnVerde, null);
    this.add(btnAzul, null);
}

public void respuestaBotones (ActionEvent ev){
    Object boton=ev.getSource();
    if(boton.equals(btnRojo)){
        indice=0;
    }else if(boton.equals(btnVerde)){
        indice=1;
    }else{
        indice=2;
    }
    repaint();
}

public void paint(Graphics g){
    g.setColor(colores[indice]);
    g.fillRect(20, 50, 100, 50);
}
}

class AccionBotones implements ActionListener{
    private BotonesApplet1 applet;
    public AccionBotones(BotonesApplet1 applet){
        this.applet=applet;
    }
    public void actionPerformed(ActionEvent ev){
        applet.respuestaBotones (ev);
    }
}

```

Respuesta a las acciones del usuario sobre un grupo de botones de radio

En la página anterior hemos visto como se define una respuesta única a la acción sobre un conjunto de tres botones. Un grupo de botones de radio, es una elección más acertada que un conjunto de tres botones, ya que en un grupo uno sólo de los botones de radio está en estado activado. Para crear un grupo de botones de radio es necesario seguir los siguientes pasos.

Los controles *Checkbox* y *CheckboxGroup*

Se crean los controles del tipo *Checkbox* y un control *CheckboxGroup*.

```
Checkbox chkRojo = new Checkbox();
Checkbox chkVerde = new Checkbox();
Checkbox chkAzul = new Checkbox();
CheckboxGroup chkGrupo=new CheckboxGroup();
```

En *init* se pone una etiqueta para identificar cada uno de los botones de radio, mediante *setLabel*,

```
chkRojo.setLabel("Rojo");
chkVerde.setLabel("Verde");
chkAzul.setLabel("Azul");
```

Se asocia cada uno de los botones de radio con su grupo, el objeto *chkGrupo* de la clase *CheckboxGroup*. Por último, se establece el botón de radio en estado activado, que se presenta inicialmente cuando aparece el applet, mediante la función miembro *setSelectedCheckbox* de la clase *CheckboxGroup*.

```
chkRojo.setCheckboxGroup(chkGrupo);
chkVerde.setCheckboxGroup(chkGrupo);
chkAzul.setCheckboxGroup(chkGrupo);
chkGrupo.setSelectedCheckbox(chkRojo);
```

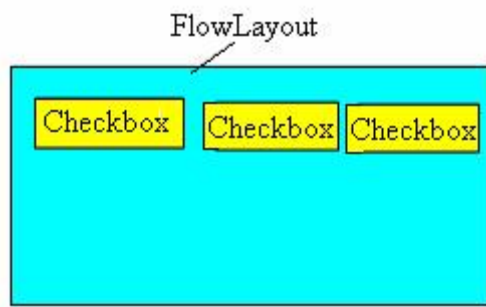
Para obtener el botón de radio que ha sido seleccionado se llama a la función miembro *getSelectedCheckbox* de la clase *CheckboxGroup*.

```
Checkbox radio=chkGrupo.getSelectedCheckbox();
```

Propósito

El applet contiene un conjunto de tres botones de radio situados en la parte superior del applet. El nombre de los botones es el de los colores primarios: azul, verde y rojo. Al activar un botón de radio se pinta un rectángulo de dicho color.

Diseño



Crear el applet, situar tres botones de radio (*Checkbox*) en la parte superior.

Cambiar sus propiedades **name** y **label** en las correspondiente hoja de propiedades

Establecer FlowLayout, como gestor de diseño del applet. Cambiar la propiedad **hgap** para separar horizontalmente los botones.

En el código fuente crear un objeto de la clase *CheckboxGroup*, asociar mediante *setCheckboxGroup* cada botón de radio al grupo. Establecer el botón de radio inicialmente activado.

Crear un array *colores* de objetos de la clase Color con los tres colores primeros: rojo, verde y azul.

```
final Color[] colores={Color.red, Color.green, Color.blue};
```

Redefinir la función *paint* para pintar un rectángulo

Respuesta a las acciones del usuario

Para responder a las acciones sobre un grupo de botones de radio se crea una clase que denominamos *ItemRadio* que implemente el interface *ItemListener*. La clase define la función *itemStateChanged*. La información acerca del suceso viene encapsulada en el objeto *ev* de la clase *ItemEvent*.

```
class ItemRadio implements ItemListener{
    private RadioApplet applet;
    public ItemRadio(RadioApplet applet){
        this.applet=applet;
    }
    public void itemStateChanged(ItemEvent ev){
        applet.funcionRadio (ev);
    }
}
```

Se crea un objeto *item* de la clase *ItemRadio* y se le pasa **this** al constructor, para inicializar el miembro dato *applet* de la clase *ItemRadio*. En la definición de la función miembro *itemStateChanged*, se llama a la función respuesta *funcionRadio*.

```
ItemRadio item=new ItemRadio(this);
```

Asociamos, mediante *addItemListener*, cada una de los botones de radio con el objeto *item* que está interesado en las acciones sobre dichos controles.

```
chkRojo.addItemListener(item);
chkVerde.addItemListener(item);
chkAzul.addItemListener(item);
```

En la definición de la función respuesta *funcionRadio*, se describe la tarea a realizar. En nuestro caso dibujar un rectángulo del color que se corresponde con el título del botón de radio.

Primero tenemos que saber sobre cual de los botones de radio hemos actuado, para ello se llama a la función miembro *setSelectedCheckbox*, de la clase *CheckboxGroup*. Posteriormente, obtenemos el índice del color del botón de radio que ha sido activado

```
public void funcionRadio (ItemEvent ev){
    Checkbox radio=chkGrupo.getSelectedCheckbox();
    if(radio.equals(chkRojo)){
        indice=0;
    }
    if(radio.equals(chkVerde)){
        indice=1;
    }
    if(radio.equals(chkAzul)){
        indice=2;
    }
    repaint();
}
```

Finalmente, se llama la función *paint* para dibujar el rectángulo pintado del color seleccionado.

```
public void paint(Graphics g){
    g.setColor(colores[indice]);
    g.fillRect(20, 50, 100, 50);
}
```

El código completo de este ejemplo, es el siguiente

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

public class RadioApplet extends Applet {
    Checkbox chkRojo = new Checkbox();
    Checkbox chkVerde = new Checkbox();
    Checkbox chkAzul = new Checkbox();
    CheckboxGroup chkGrupo=new CheckboxGroup();
    FlowLayout flowLayout1 = new FlowLayout();
    final Color[] colores={Color.red, Color.green, Color.blue};
    int indice;

    public void init() {
        chkRojo.setLabel("Rojo");
        chkVerde.setLabel("Verde");
        chkAzul.setLabel("Azul");
    }
}
```

```

chkRojo.setCheckboxGroup(chkGrupo);
chkVerde.setCheckboxGroup(chkGrupo);
chkAzul.setCheckboxGroup(chkGrupo);
chkGrupo.setSelectedCheckbox(chkRojo);

ItemRadio item=new ItemRadio(this);
chkRojo.addItemListener(item);
chkVerde.addItemListener(item);
chkAzul.addItemListener(item);
FlowLayout1.setHgap(10);
this.setLayout(FlowLayout1);
this.add(chkRojo, null);
this.add(chkVerde, null);
this.add(chkAzul, null);
}

public void funcionRadio (ItemEvent ev){
    Checkbox radio=chkGrupo.getSelectedCheckbox();
    indice=0;
    if(radio.equals(chkRojo)){
        indice=0;
    }
    if(radio.equals(chkVerde)){
        indice=1;
    }
    if(radio.equals(chkAzul)){
        indice=2;
    }
    repaint();
}

public void paint(Graphics g){
    g.setColor(colores[indice]);
    g.fillRect(20, 50, 100, 50);
}
}

class ItemRadio implements ItemListener{
    private RadioApplet applet;
    public ItemRadio(RadioApplet applet){
        this.applet=applet;
    }
    public void itemStateChanged(ItemEvent ev){
        applet.funcionRadio (ev);
    }
}

```