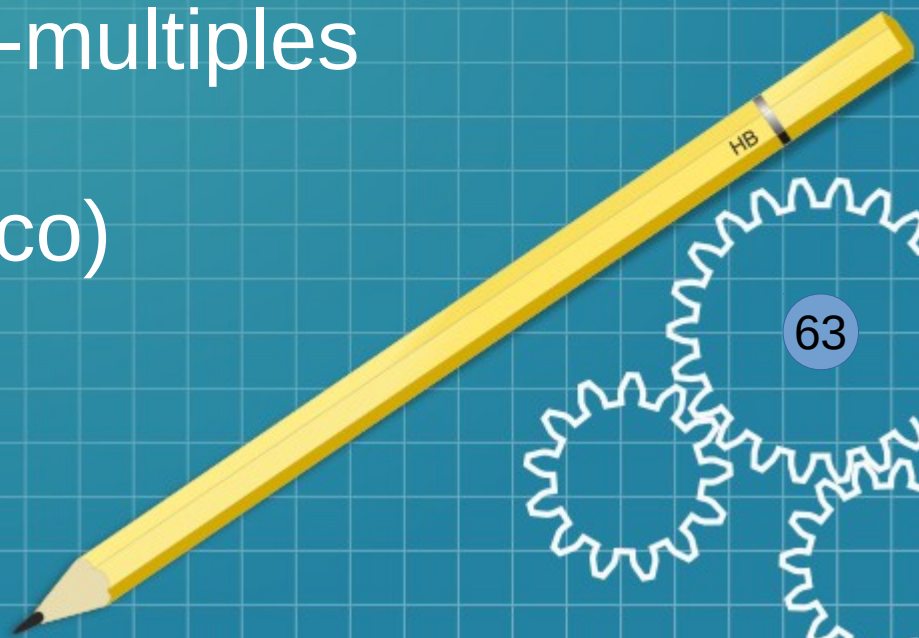
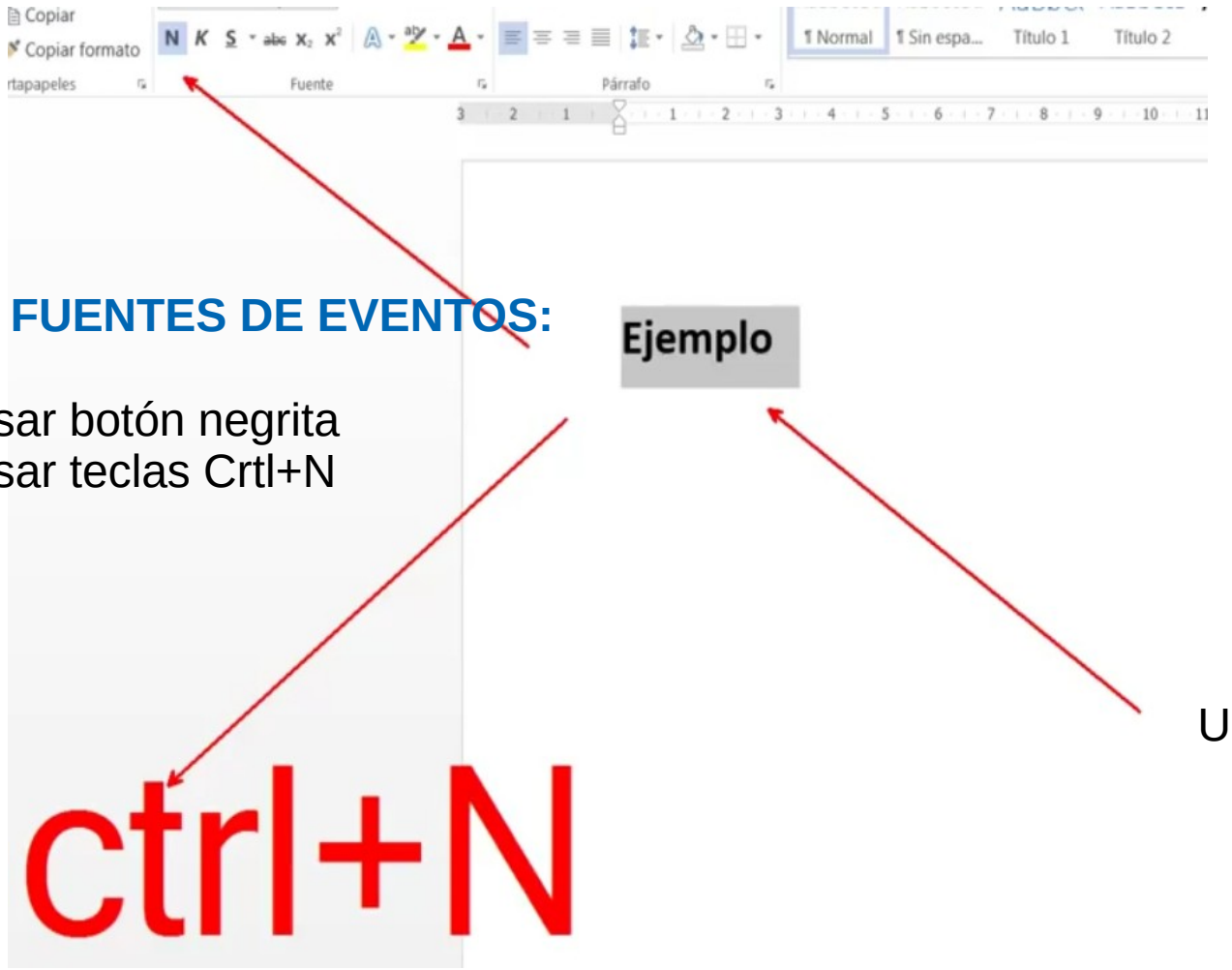
Java

Swing – Eventos -multiples

(Toni Franco)



Fuentes de eventos múltiples



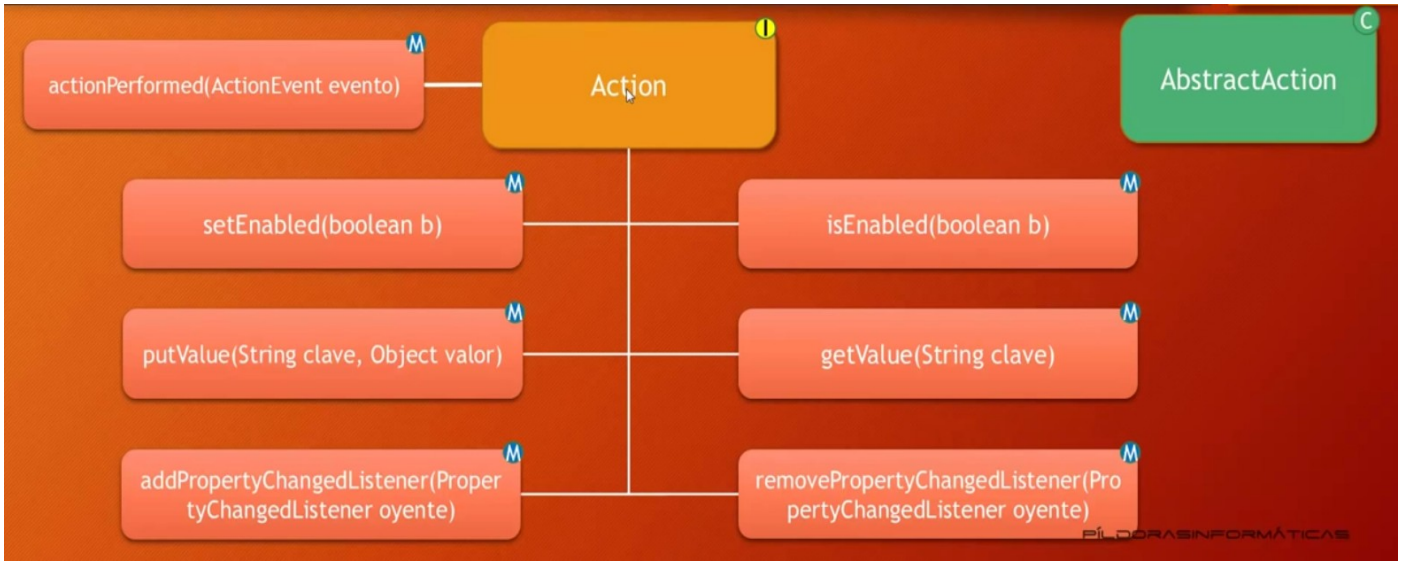
DOS FUENTES DE EVENTOS:

- Pulsar botón negrita
- Pulsar teclas Crtl+N

Un sólo oyente

Toni Franco

Eventos multiples usando Action



Antes haciamos:

```
botonAzul.addActionListener(Azul);
```

El método que se ejecutaba al ocurrir un evento lo tiene el listener (oyente)

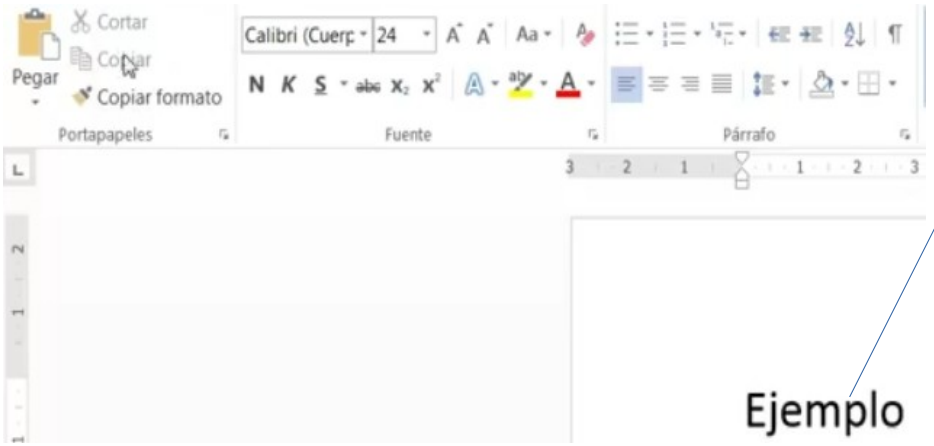
```
private class ColorFondo implements ActionListener{

    public ColorFondo(Color c){

        colorDeFondo=c;
    }

    public void actionPerformed(ActionEvent e){

        setBackground(colorDeFondo);
    }
```

Ahora usamos Action que:

Hereda actionPerformed y es una interface que no tiene una clase adapter. En cambio hay una clase AbstractAction que hace algo parecido para no implementar todos los métodos.
Cuando se genera un evento, se llamará al método actionPerformed de una Action, es decir,que Action puede actuar de oyente.

Toni Franco

Ejemplo



Si el objeto oyente («Ejemplo») Está seleccionado, el botón Copiar (evento fuente) esta activado

El objeto oyente es quien controla la activación

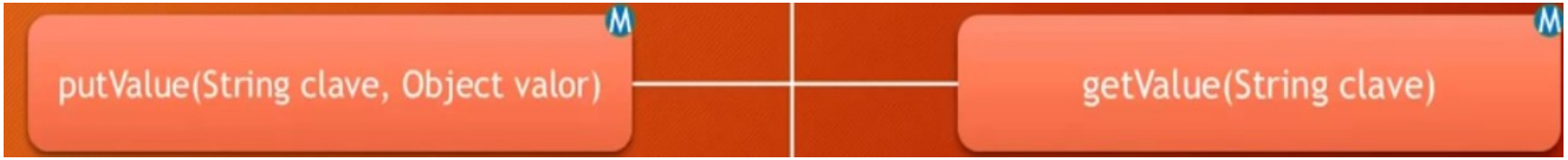
Con estos dos metodos se selecciona el texto y se activa el botón de copiar:

addPropertyCangedListener(oyente)
removePropertyCangedListener(oyente)

Con los métodos siguientes activamos o comprobamos si està activo el botón:

setEnabled(true/false)
isEnabled()

AbstractAction



Almacenar / recuperar datos que deseemos estructurandolos usando una clave que lo identifica

Es como una clases Adapter que hereda los métodos ActionPerformed, setEnabled, isEnabled, getValue, PutValue, ...

Constructores de un botón

Constructor Summary

Constructors

Constructor and Description
`JButton()` Creates a button with no set text or icon.
`JButton(Action a)` Creates a button where properties are taken from the `Action` supplied.
`JButton(Icon icon)` Creates a button with an icon.
`JButton(String text)` Creates a button with text.
`JButton(String text, Icon icon)` Creates a button with initial text and an icon.

Ahora vamos a usar este constructor, que puede recibir una «Action»

Hasta ahora hemos usado estos contructores

Opciones al crear un botón

En la interface Action podemos ver todo lo que se puede usar en el constructor para almacenarlo con putValue. De este modo se modifica el botón creado.

Fields

Modifier and Type	Field and Description
`static String`	`ACCELERATOR_KEY` The key used for storing a `KeyStroke` to be used as the accelerator for the action.
`static String`	`ACTION_COMMAND_KEY` The key used to determine the command `String` for the `ActionEvent` that will be created when an `Action` is going to be notified as the result of residing in a `Keymap` associated with a `JComponent`.
`static String`	`DEFAULT` Not currently used.
`static String`	`DISPLAYED_MNEMONIC_INDEX_KEY` The key used for storing an `Integer` that corresponds to the index in the text (identified by the `NAME` property) that the decoration for a mnemonic should be rendered at.
`static String`	`LARGE_ICON_KEY` The key used for storing an `Icon`.
`static String`	`LONG_DESCRIPTION` The key used for storing a longer `String` description for the action, could be used for context-sensitive help.
`static String`	`MNEMONIC_KEY` The key used for storing an `Integer` that corresponds to one of the `KeyEvent` key codes.
`static String`	`NAME` The key used for storing the `String` name for the action, used for a menu or button.
`static String`	`SELECTED_KEY` The key used for storing a `Boolean` that corresponds to the selected state.
`static String`	`SHORT_DESCRIPTION` The key used for storing a short `String` description for the action, used for tooltip text.
`static String`	`SMALL_ICON` The key used for storing a small `Icon`, such as `ImageIcon`.

Ejemplo AbstractAction

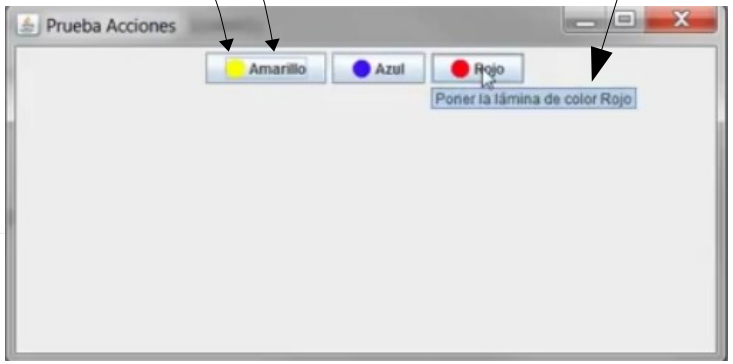
```
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        MarcoAccion marco=new MarcoAccion();

        marco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        marco.setVisible(true);

    }

}

class MarcoAccion extends JFrame{

    public MarcoAccion(){

        setTitle("Prueba Acciones");

        setBounds(600,350,600,300);

        PanelAccion lamina=new PanelAccion();

        add(lamina);

    }

}
class PanelAccion extends JPanel{

        public PanelAccion(){

         AccionColor accionAmarillo=new AccionColor("Amarillo", new ImageIcon("src/graficos/bola_amarilla.gif"), Color.yellow);
          AccionColor accionAzul=new AccionColor("Azul", new ImageIcon("src/graficos/bola_azul.gif"), Color.blue);
         AccionColor accionRojo=new AccionColor("Rojo", new ImageIcon("src/graficos/bola_roja.gif"), Color.red);
        add(new JButton(accionAmarillo));

        add(new JButton(accionAzul));

        add(new JButton(accionRojo));

    }
```

Esto es crear
una Action que es un oyente (listener)
con unas propiedades que
las puede usar un botón en su creación

Crea el botón asociandole
el oyente

```
class AccionColor extends AbstractAction{

    public AccionColor(String nombre, Icon icono, Color color_boton){

        putValue(Action.NAME, nombre);

        putValue(Action.SMALL_ICON, icono);

        putValue(Action.SHORT_DESCRIPTION, "Poner la lámina de color " + nombre);

        putValue("color_de_fondo", color_boton);

    }

   public void actionPerformed(ActionEvent e) {

       Color c=(Color)getValue("color_de_fondo");

       setBackground(c);

   }


}

}
```

Tooltip



Propiedad del objeto Action

Otro ej: Varios oyentes

```
class Lamina_Principal extends JPanel{

    public Lamina_Principal(){

        JButton boton_nuevo=new JButton("Nuevo");

        add(boton_nuevo);

        boton_cerrar=new JButton("Cerrar todo");

        add(boton_cerrar);

        OyenteNuevo miOyente=new OyenteNuevo();

        boton_nuevo.addActionListener(miOyente);


    }

    private class OyenteNuevo implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub

            Marco_Emergente marco=new Marco_Emergente(boton_cerrar);

            marco.setVisible(true);

        }
    }

     JButton boton_cerrar;

    }
```

Cada vez que pulsa el botón crea una ventana que tiene un oyente para esperar el evento de cierre de la misma

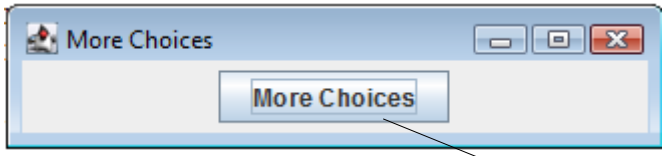
```
class Marco_Emergente extends JFrame{

    public Marco_Emergente(JButton boton_de_principal){

        contador++;

        setTitle("Ventana " + contador);

        setBounds(40*contador,40*contador, 300, 150);

        CierraTodos oyenteCerrar=new CierraTodos();

        boton_de_principal.addActionListener(oyenteCerrar);


    }

    private class CierraTodos implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub

            dispose();

        }


    }
    private static int contador=0;

}
```

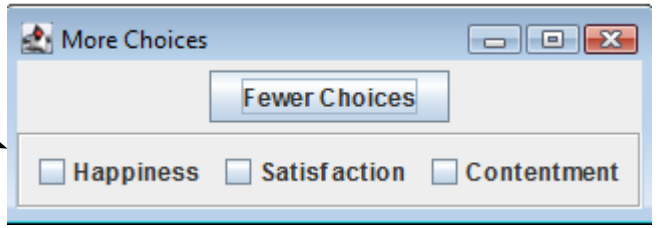
Cierra ventana

Toni Franco

RemoveActionListener

Un botón hace dos cosas





Tenemos dos jPanels. Un principal y otro con opciones que lo añadimos dentro del principal

```
class ChoicesPanel extends JPanel {
  ChoicesPanel() {
    setBorder(BorderFactory.createEtchedBorder());
    add(new JCheckBox("Happiness"));
    add(new JCheckBox("Satisfaction"));
    add(new JCheckBox("Contentment"));
  }
}
```

```
less = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    cp.remove(moreOrLessPanel); // Quita el jPanel de las opciones del JPanel principal
    pack();
    Boton.setText("More Choices");
    Boton.removeActionListener(less);
    Boton.addActionListener(more);
  }
};
more = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    cp.add(BorderLayout.SOUTH, moreOrLessPanel);  // Añade el JPanel de las opciones dentro de jPanel principal
    pack();
    Boton.setText("Fewer Choices");
    Boton.removeActionListener(more);
    Boton.addActionListener(less);
  }
};
bp.add(Boton = new JButton("More Choices"));
// Initial state is to add more choices
Boton.addActionListener(more);

moreOrLessPanel = new ChoicesPanel(); // Crea el jpanel con las opciones
```

Añade el botón al Jpanel
Y activa un listener

Quita el listener y
Asocia otro

Quita el listener y asocia
otro

Se entrega el código QuitarOyente.java

Toni Franco

Cuidado con anidar

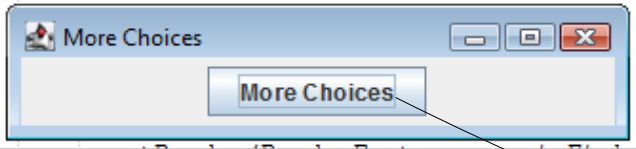
Modifica el código de antes creando un MouseListener para cuando entras dentro de la ventana principal. Observa como se mezclan los eventos.

```
    more = new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        cp.add(BorderLayout.SOUTH, moreOrLessPanel);  // Añade el
        pack();
        Boton.setText("Fewer Choices");
        Boton.removeActionListener(more);
        Boton.addActionListener(less);
        System.out.println("has hecho clic1");
      }
    };
    bp.add(Boton = new JButton("More Choices"));
    // Initial state is to add more choices
    Boton.addActionListener(more);
    moreOrLessPanel = new ChoicesPanel(); // Crea el jpanel con las opciones

    addMouseListener(new OyentePrincipal());

    // Finally a frame closer
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        setVisible(false);
        dispose();
        System.exit(0);
      }
    });

    pack();
  }

  class OyentePrincipal extends MouseAdapter{
      public void mouseEntered(MouseEvent e){
          System.out.println("Has entrado "+e.getSource());
      }
  }
```

Puedes controlarlo con removeActionListener u otras acciones...

Funcionamiento

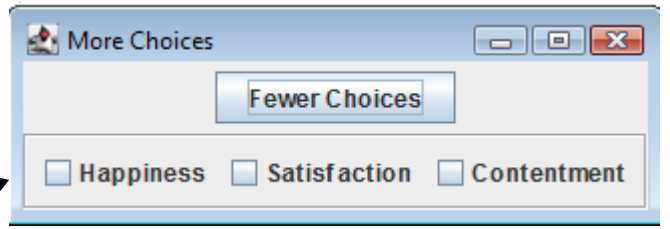
```
class OyentePrincipal extends MouseAdapter{
    public void mouseEntered(MouseEvent e){
        System.out.println("Has entrado ");
    }
     public void mouseMoved(MouseEvent e){
        System.out.println("mov");
    }
    public void mouseExited(MouseEvent e){
        System.out.println("Ha salido ");
    }
    public void mousePressed(MouseEvent e){
        System.out.println("Ha pulsado ");
    }
}

/**
 * The panel that appears when you press More
                                          be a se
```



emoveListener.OyenteAnidado > OyentePrincipal > mousePress

- GUI01 (run)

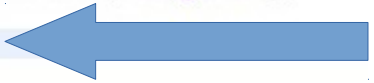
```
Ha salido
has hecho clic2
has hecho clic1
has hecho clic2
Has entrado
Ha pulsado
Ha salido
Has entrado
Ha salido
Has entrado
Ha salido
```



Al pulsar el botón aparece el jpanel con las opciones

Refrescar un jPanel

```
import java.awt.event.ActionEvent;
public class RefrescarPanel extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private Panel panel;
    public static void main(String[] args) {
            RefrescarPanel frame = new RefrescarPanel();
            frame.setVisible(true);
    }
    public RefrescarPanel() {
        setTitle("Refrescar Fondo");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        panel = new Panel();//Creo el Panel
        panel.setUrl("/imagenes/imagen1.jpg");//establezco el valor de url
        TitledBorder tb=BorderFactory.createTitledBorder("Imagen 1");
        panel.setBorder(tb);//borde
        panel.setBounds(10, 11, 414, 207);//posición
        contentPane.add(panel);//agrego a contenPane

        JButton btnNewButton = new JButton("Cambiar Fondo");
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                panel.setUrl("/imagenes/imagen2.jpg");//establezco el valor de url
                TitledBorder tb=BorderFactory.createTitledBorder("Imagen 2");
                panel.setBorder(tb);//borde
                panel.updateUI();}//Actualizo el Panel.
        });
        btnNewButton.setBounds(10, 229, 414, 23);
        contentPane.add(btnNewButton);
    }
}
```



Toni Franco