33 - Swing - JButton

Listado completo de tutoriales

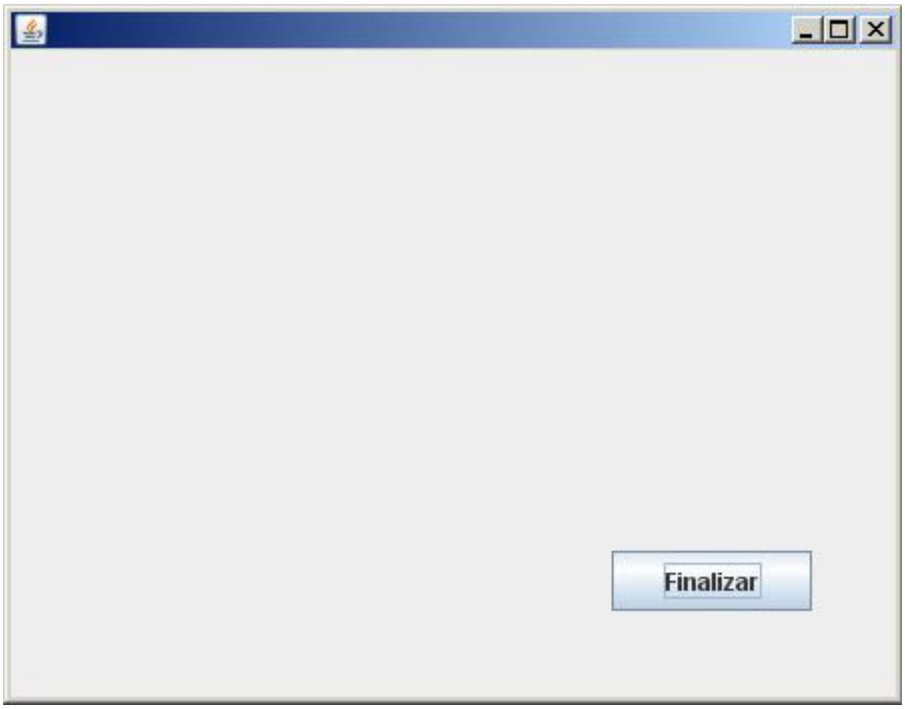
El tercer control visual de uso muy común es el que provee la clase JButton. Este control visual muestra un botón.

El proceso para añadir botones a un control JFrame es similar a añadir controles de tipo JLabel.

Ahora veremos la captura de eventos con los controles visuales. Uno de los eventos más comunes es cuando hacemos clic sobre un botón.
Java implementa el concepto de interfaces para poder llamar a métodos de una clase existente a una clase desarrollada por nosotros.

Problema 1:

Confeccionar una ventana que muestre un botón. Cuando se presione finalizar la ejecución del programa Java.



Programa:

Ver video

```
import javax.swing.*;
import java.awt.event.*;
public class Formulario extends JFrame implements Act
    JButton boton1;
    public Formulario() {
```

```
        setLayout(null);
        boton1=new JButton("Finalizar");
        boton1.setBounds(300,250,100,30);
        add(boton1);
        boton1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            System.exit(0);
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,450,350);
        formulario1.setVisible(true);
        formulario1.setDefaultCloseOperation(JFrame.E
    }
}
```

La mecánica para atrapar el clic del objeto de la clase JButton se hace mediante la implementación de una interface. Una interface es un protocolo que permite la comunicación entre dos clases. Una interface contiene uno o más cabecera de métodos, pero no su implementación. Por ejemplo la interface ActionListener tiene la siguiente estructura:

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

Luego las clases que implementen la interface ActionListener deberán especificar el algorítmo del método actionPerformed.

Mediante el concepto de interfaces podemos hacer que desde la clase JButton se puede llamar a un método que implementamos en nuestra clase.

Para indicar que una clase implementará una interface lo hacemos en la declaración de la clase con la sintaxis:

```
public class Formulario extends JFrame implements ActionListener {
```

Con esto estamos diciendo que nuestra clase implementa la interface ActionListener, luego estamos obligados a codificar el método actionPerformed.

Definimos un objeto de la clase JButton:

```
JButton boton1;
```

En el constructor creamos el objeto de la clase JButton y mediante la llamada del método addActionListener le pasamos la referencia del objeto de la clase JButton utilizando la palabra clave this (this almacena la dirección de memoria donde se almacena el objeto de la clase JFrame, luego mediante dicha dirección podemos llamar al método actionPerformed):

```
public Formulario() {
    setLayout(null);
    boton1=new JButton("Finalizar");
    boton1.setBounds(300,250,100,30);
    add(boton1);
    boton1.addActionListener(this);
}
```

El método actionPerformed (este método de la interface ActionListener debe implementarse obligatoriamente, en caso de no codificarlo o equivocarnos en su nombre aparecerá un mensaje de error en tiempo de compilación de nuestro programa.

El método actionPerformed se ejecutará cada vez que hagamos clic sobre el objeto de la clase JButton.

La interface ActionListener y el objeto de la clase ActionEvent que llega como parámetro están definidos en el paquete:

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

Es decir que cada vez que se presiona el botón desde la clase JButton se llama al método actionPerformed y recibe como parámetro un objeto de la clase ActionEvent.

En el método actionPerformed mediante el acceso al método getSource() del objeto que llega como parámetro podemos analizar que botón se presionó:

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource()==boton1) {
        System.exit(0);
    }
}
```

Si se presionón el boton1 luego el if se verifica verdadero y por lo tanto llamando al método exit de la clase System se finaliza la ejecución del programa.

La main no varía en nada con respecto a problemas anteriores.

Problema 2:

Confeccionar una ventana que contenga tres objetos de la clase JButton con las etiquetas "1", "2" y "3". Al presionarse cambiar el título del JFrame indicando cuál botón se presionó.

Programa:

Ver video

```
import javax.swing.*;
import java.awt.event.*;
public class Formulario extends JFrame implements Act
    private JButton boton1,boton2,boton3;
    public Formulario() {
        setLayout(null);
        boton1=new JButton("1");
        boton1.setBounds(10,100,90,30);
        add(boton1);
        boton1.addActionListener(this);
        boton2=new JButton("2");
        boton2.setBounds(110,100,90,30);
        add(boton2);
        boton2.addActionListener(this);
        boton3=new JButton("3");
        boton3.setBounds(210,100,90,30);
        add(boton3);
        boton3.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            setTitle("boton 1");
        }
        if (e.getSource()==boton2) {
            setTitle("boton 2");
        }
```

```
        if (e.getSource()==boton3) {
            setTitle("boton 3");
        }
    }

    public static void main(String[] ar){
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,350,200);
        formulario1.setVisible(true);
        formulario1.setDefaultCloseOperation(JFrame.E
    }
}
```

Debemos declarar 3 objetos de la clase JButton:

```
    private JButton boton1,boton2,boton3;
```

En el constructor creamos los tres objetos de la clase JButton y los ubicamos dentro del control JFrame (también llamamos al método addActionListener para enviarle la dirección del objeto de la clase Formulario):

```
    public Formulario() {
        setLayout(null);
        boton1=new JButton("1");
        boton1.setBounds(10,100,90,30);
        add(boton1);
        boton1.addActionListener(this);
        boton2=new JButton("2");
        boton2.setBounds(110,100,90,30);
        add(boton2);
        boton2.addActionListener(this);
        boton3=new JButton("3");
        boton3.setBounds(210,100,90,30);
        add(boton3);
        boton3.addActionListener(this);
    }
```

Cuando se presiona alguno de los tres botones se ejecuta el método actionPerformed y mediante tres if verificamos cual de los botones se presionó:

```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            setTitle("boton 1");
        }
        if (e.getSource()==boton2) {
            setTitle("boton 2");
        }
        if (e.getSource()==boton3) {
```

```
            setTitle("boton 3");
        }
    }
```

Según el botón presionado llamamos al método setTitle que se trata de un método heredado de la clase JFrame y que tiene por objetivo mostrar un String en el título de la ventana.

Problema propuesto

1. Disponer dos objetos de la clase JButton con las etiquetas: "varón" y "mujer", al presionarse mostrar en la barra de títulos del JFrame la etiqueta del botón presionado.
   Ver video

Solución

Retornar