

Curso de Java a distancia

Clase 11: Swing - JFrame

La componente básica que requerimos cada vez que implementamos una interfaz visual con la librería Swing es la clase JFrame. Esta clase encapsula una Ventana clásica de cualquier sistema operativo con entorno gráfico (Windows, OS X, Linux etc.)

Hemos dicho que esta clase se encuentra en el paquete javax.swing y como generalmente utilizamos varias clases de este paquete luego para importarla utilizamos la sintaxis:

```
import javax.swing.*;
```

Podemos hacer una aplicación mínima con la clase JFrame:

```
import javax.swing.JFrame;

public class Formulario {

    public static void main(String[] ar) {

        JFrame f=new JFrame();

        f.setBounds(10,10,300,200);

        f.setVisible(true);

    }

}
```

Como vemos importamos la clase JFrame del paquete javax.swing:

```
import javax.swing.JFrame;
```

y luego en la main definimos y creamos un objeto de la clase JFrame (llamando luego a los métodos setBounds y setVisible):

```
public static void main(String[] ar) {  
  
    JFrame f=new JFrame();  
  
    f.setBounds(10,10,300,200);  
  
    f.setVisible(true);  
  
}
```

Pero esta forma de trabajar con la clase JFrame es de poca utilidad ya que rara vez necesitemos implementar una aplicación que muestre una ventana vacía.

Lo más correcto es plantear una clase que herede de la clase JFrame y extienda sus responsabilidades agregando botones, etiquetas, editores de línea etc.

Entonces la estructura básica que emplearemos para crear una interfaz visual será:

Programa:

```
import javax.swing.*.*;  
  
public class Formulario extends JFrame{  
  
    public Formulario() {  
  
        setLayout(null);  
  
    }  
  
  
    public static void main(String[] ar) {  
  
        Formulario formulario1=new Formulario();  
  
        formulario1.setBounds(10,20,400,300);  
  
        formulario1.setVisible(true);  
  
    }  
  
}
```

Importamos el paquete donde se encuentra la clase JFrame:

```
import javax.swing.*.*;
```

Planteamos una clase que herede de la clase JFrame:

```
public class Formulario extends JFrame{
```

En el constructor indicamos que ubicaremos los controles visuales con coordenadas absolutas mediante la desactivación del Layout heredado (más adelante veremos otras formas de ubicar los controles visuales dentro del JFrame):

```
    public Formulario() {  
  
        setLayout(null);  
  
    }
```

En la main creamos un objeto de la clase y llamamos a los métodos setBounds y setVisible:

```
    public static void main(String[] ar) {  
  
        Formulario formulario1=new Formulario();  
  
        formulario1.setBounds(10,20,400,300);  
  
        formulario1.setVisible(true);  
  
    }
```

El método setBounds ubica el JFrame (la ventana) en la columna 10, fila 20 con un ancho de 400 píxeles y un alto de 300.

Debemos llamar al método setVisible y pasarle el valor true para que se haga visible la ventana.

Problemas propuestos

1. Crear una ventana de 1024 píxeles por 800 píxeles. Luego no permitir que el operador modifique el tamaño de la ventana. Sabiendo que hacemos visible al JFrame llamando la método setVisible pasando el valor true, existe otro método llamado setResizable que también requiere como parámetro un valor true o false.

Solución

```
import javax.swing.*;

public class Formulario extends JFrame{

    Formulario() {

        setLayout(null);

    }

    public static void main(String[] ar) {

        Formulario formulario1=new Formulario();


        formulario1.setBounds(0,0,1024,800);

        formulario1.setResizable(false);

        formulario1.setVisible(true);

    }

}
```



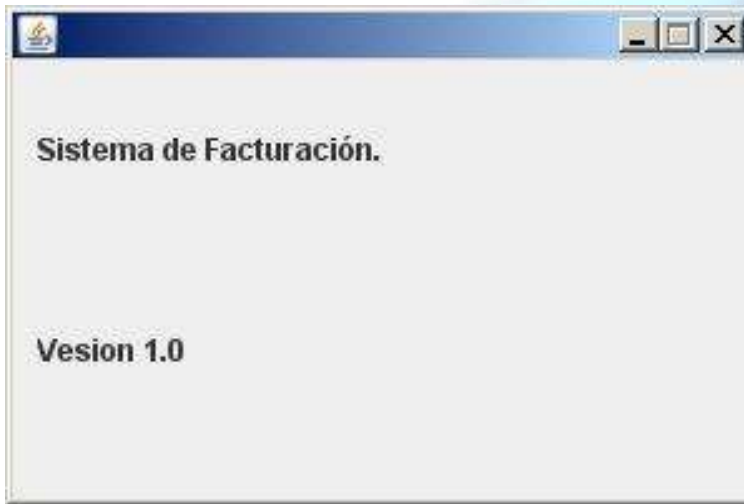
Swing - JLabel

Veamos la segunda componente de la librería swing llamada JLabel.

La clase JLabel nos permite mostrar básicamente un texto.

Problema 1:

Confeccionar una ventana que muestre el nombre de un programa en la parte superior y su número de versión en la parte inferior. No permitir modificar el tamaño de la ventana en tiempo de ejecución.



Programa:

```
import javax.swing.*;

public class Formulario extends JFrame {

    private JLabel label1,label2;

    public Formulario() {

        setLayout(null);

        label1=new JLabel("Sistema de Facturación.");

        label1.setBounds(10,20,300,30);

        add(label1);

        label2=new JLabel("Vesion 1.0");

        label2.setBounds(10,100,100,30);
```

```
add(label2);  
  
}
```

```
public static void main(String[] ar) {  
  
    Formulario formulario1=new Formulario();  
  
    formulario1.setBounds(0,0,300,200);  
  
    formulario1.setResizable(false);  
  
    formulario1.setVisible(true);  
  
}  
}
```

Importamos el paquete javax.swing donde se encuentran definidas las clase JFrame y JLabel:

```
import javax.swing.*;
```

Heredamos de la clase de JFrame:

```
public class Formulario extends JFrame {
```

Definimos dos atributos de la clase JLabel:

```
private JLabel label1,label2;
```

En el constructor creamos las dos JLabel y las ubicamos llamando al método setBounds, no hay que olvidar de llamar al método add que añade la JLabel al JFrame:

```
public Formulario() {  
  
    setLayout(null);  
  
    label1=new JLabel("Sistema de Facturación.");  
  
    label1.setBounds(10,20,300,30);  
  
    add(label1);  
  
    label2=new JLabel("Vesion 1.0");
```

```
label2.setBounds(10,100,100,30);

add(label2);

}
```

Por último en la main creamos un objeto de la clase que acabamos de codificar, llamamos al `setBounds` para ubicar y dar tamaño al `JFrame`, llamamos al método `setResizable` pasando el valor `false` para no permitir modificar el tamaño del `JFrame` en tiempo de ejecución y finalmente llamamos al método `setVisible` para que se visualice el `JFrame`:

```
public static void main(String[] ar) {

    Formulario formulario1=new Formulario();

    formulario1.setBounds(0,0,300,200);

    formulario1.setResizable(false);

    formulario1.setVisible(true);

}
```

Problemas propuestos

1. Crear tres objetos de la clase `JLabel`, ubicarlos uno debajo de otro y mostrar nombres de colores.

Solución

```
import javax.swing.*;

public class Formulario extends JFrame {

    private JLabel label1,label2,label3;

    public Formulario() {

        setLayout(null);

        label1=new JLabel("Rojo");

        label1.setBounds(10,20,100,30);

        add(label1);

        label2=new JLabel("Verde");

        label2.setBounds(10,60,100,30);

        add(label2);

        label3=new JLabel("Azul");

        label3.setBounds(10,100,100,30);

        add(label3);

    }

    public static void main(String[] ar) {

        Formulario formulario1=new Formulario();

        formulario1.setBounds(0,0,300,200);
```



```
formulario1.setVisible(true);  
  
}  
  
}
```



Swing - JButton

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

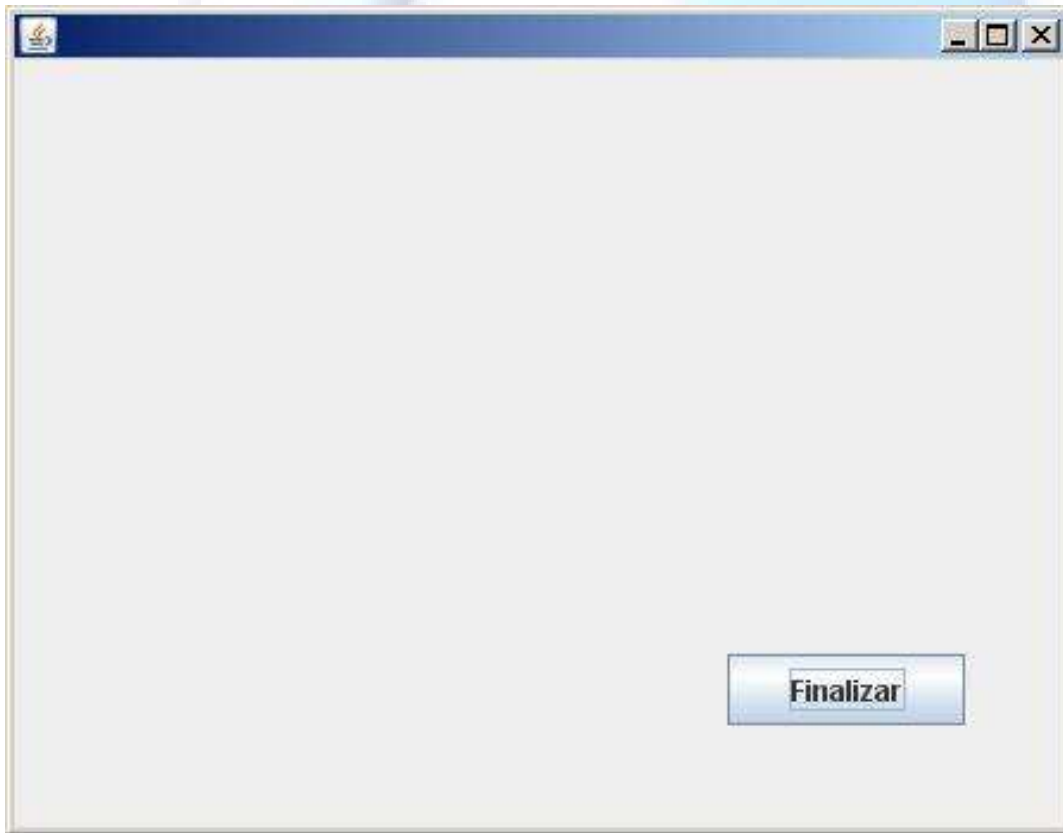
El proceso para añadir botones a un control JFrame es similar a añadir controles de tipo JLabel.

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:

```
import javax.swing.*;

import java.awt.event.*;

public class Formulario extends JFrame implements ActionListener {
```

```

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);

}

}

```

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 algoritmo 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:

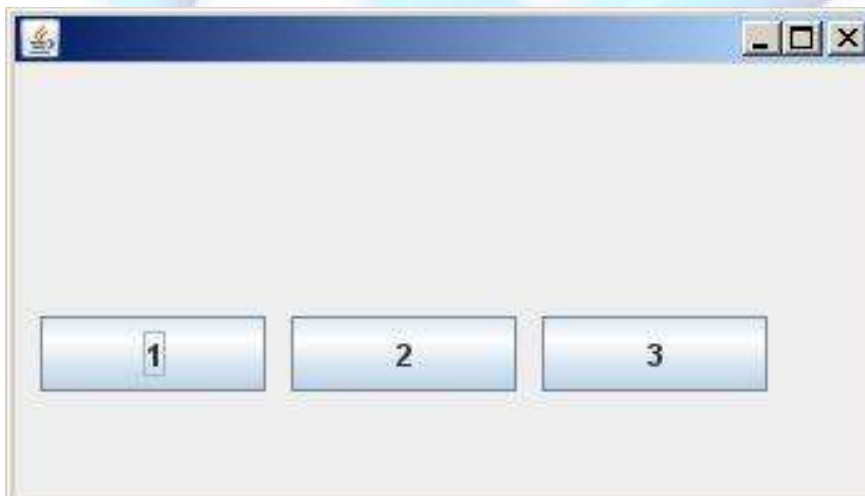
```
public void actionPerformed(ActionEvent e) {  
  
    if (e.getSource()==boton1) {  
  
        System.exit(0);  
  
    }  
  
}
```

Si se presionó 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:

```
import javax.swing.*;

import java.awt.event.*;

public class Formulario extends JFrame implements ActionListener{

    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);

}

}

```

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.

Problemas propuestos

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.



Solución

```
import javax.swing.*;

import java.awt.event.*;

public class Formulario extends JFrame implements ActionListener{

    private JButton boton1, boton2;

    public Formulario() {

        setLayout(null);

        boton1=new JButton("Varón");

        boton1.setBounds(10,10,100,30);

        boton1.addActionListener(this);

        add(boton1);

        boton2=new JButton("Mujer");

        boton2.setBounds(10,70,100,30);

        boton2.addActionListener(this);

        add(boton2);

    }

    public void actionPerformed(ActionEvent e) {

        if (e.getSource()==boton1) {

            setTitle("Varón");

        }

        if (e.getSource()==boton2) {
```

```
        setTitle("Mujer");  
    }  
}
```

```
public static void main(String[] ar) {  
    Formulario formulario1=new Formulario();  
    formulario1.setBounds(0,0,130,140);  
    formulario1.setVisible(true);  
}  
}
```



Muchas gracias hasta la próxima clase.

Alsina 16 [B1642FNB] San Isidro | Pcia. De Buenos Aires |Argentina |

TEL.: [011] 4742-1532 o [011] 4742-1665 |

www.institutosanisidro.com.ar info@institutosanisidro.com.ar