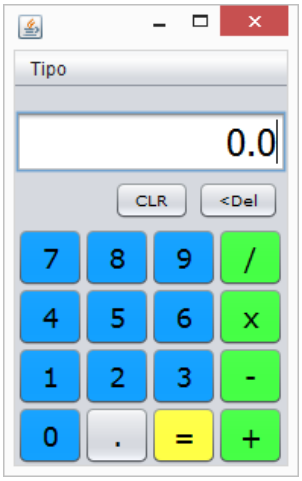
TAP Labo 7/11/2014



Vamos a usar (o generar) nuestro interfaz de calculadora para hacer una versión que funcione y con cierta corrección en su arquitectura.

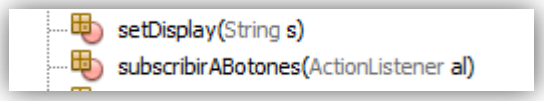
La aplicación arrancará de una clase de control que se limita a poner en marcha el GUI (instanciar un objeto de nuestra clase que hereda JFrame) y luego arrancar el controlador de la calculadora. El código es un tanto "truculento", por lo que se da en la imagen de más abajo. Ya lo explicaremos más adelante.

(se deberá adecuar como corresponda: situar en el paquete definido, y llamar a las clases como se desee; en este caso el GUI es de clase "CalcuGUI" y el controlador de clase "CalculadoraElectronica")

Al controlador se le aporta el GUI para que pueda gestionar la suscripción a sus eventos y aportarle información a mostrar .

```
public final class Calculadora {

    private static CalcuGUI gui;

    public static void main(String args[]) throws Exception {
        try {
             javax.swing.UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
        }catch (Exception ignore) {}

        java.awt.EventQueue.invokeAndWait(new Runnable() {
           @Override
            public void run() {
                (gui= new CalcuGUI()).setVisible(true);
            }
        });

        new CalculadoraElectronica(gui);
    }
}
```

Antes de ver el controlador, retoquemos el JFrame: Quitamos (o no ponemos) ninguna atención a eventos de los botones. Dejaremos que de eso se encargue el controlador. El JFrame será sólo "fachada" (bueno, y lanzador de eventos). Será el controlador quien se encargue de atender eventos y decirle al JFrame qué poner en el display, y para ello el JFrame ha de proporcionarle estos dos métodos:



que son el **PROBLEMA 1**

Pistas:

"set Display" es muy fácil, no necesita pistas...

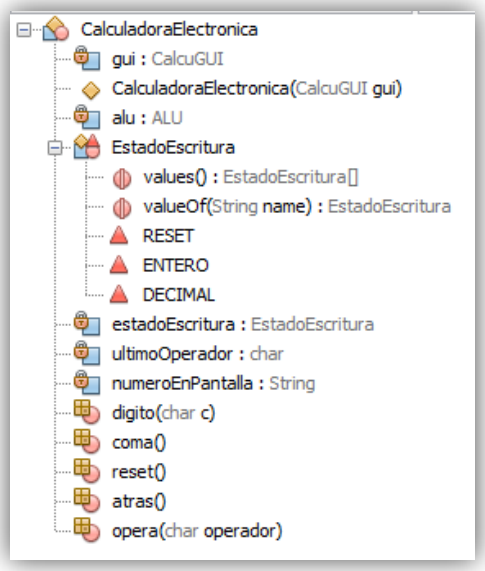
"suscribirABotones" debe recorrer todos los paneles que contengan botones para que a todos los botones contenidos se les suscriba el objeto que le llega como parámetro. Métodos a usar:

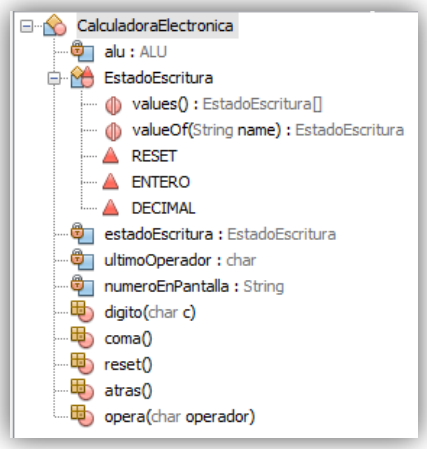
jp.getComponentCount() al JPanel (jp) podemos pedirle el número de componentes que tiene;
jp.getComponent(i) al JPanel (jp) podemos pedirle el componente i-ésimo
b.addActionListener(al) a un botón b podemos suscribirle un ActionListener (al)

De este modo podemos centrarnos en el controlador ("CalculadoraElectronica" en la imagen anterior).
Por mantener la estructura vista en clase, con una ALU muy simple, podemos mantener aquella:

```java
/**
 * Arithmetic-Logical Unit (en realidad sólo aritmética y muy sencilla).
 *
 * Arquitectura basada en acumulador (como las primeras ALUs)
 * Sólo ejecuta las cuatro operaciones aritméticas básicas.
 * Trabaja con aritmética decimal en base 10.
 *
 * @author German
 * @version 1.0 (oct 2012)
 */
public class ALU {
    private double acumulador=0.0;
    private double registro=0.0;

    double getAcumulador() { return acumulador; }
    void   clear()         { acumulador = 0.0; }
    void   setRegistro(double registro) { this.registro = registro; }
    double suma()          { return acumulador+=registro; }
    double resta()         { return acumulador-=registro; }
    double multiplica()    { return acumulador*=registro; }
    double divide()        { return acumulador/=registro; }
}
```

Vamos con el controlador: puede basarse en el visto en clase, que tenía la estructura de la imagen de la derecha, pero ahora le añadimos el constructor que le permite recibir la referencia al GUI (y el campo correspondiente) como se muestra en la imagen de abajo.





Como puede verse, la enumeración interna que pusimos para controlar los estados del número en el Display tiene un par de métodos que no escribimos, y es que se añaden "de oficio", no es preciso escribirlas.

Ahora las llamadas no le llegarán desde el GUI. Puede encargarse él mismo de escuchar los eventos o encargárselo a otro objeto y que sea este quien llame a los métodos. Es más "elegante" lo segundo, y puede hacerse generando un objeto de una clase anónima en el constructor. Este constructor quedará así:

```java
public CalculadoraElectronica(CalcuGUI gui) {
    this.gui=gui;
    gui.subscribirABotones(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent evt) {...23 lines }
    });
}
```

Para escribir el método "actionPerformed" tendremos que "descubrir" cuál ha sido el botón pulsado, cosa que podemos hacer "mirando dentro" del ActionEvent que recibimos... lo que es el **PROBLEMA 2**

Pistas: podremos descubrir quien ha sido el componente gráfico origen ("source") , que será un botón, y a este pedirle el texto que muestra en el GUI, con lo cual decidir a qué método llamar.