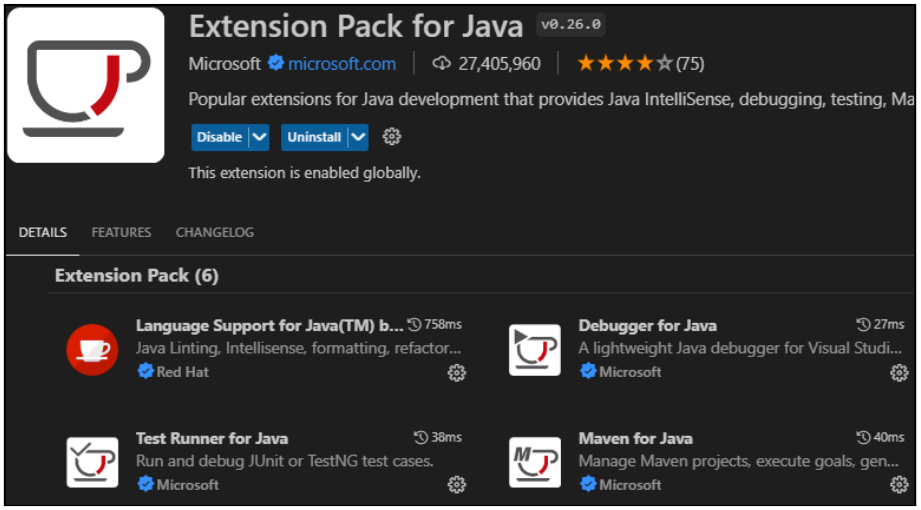
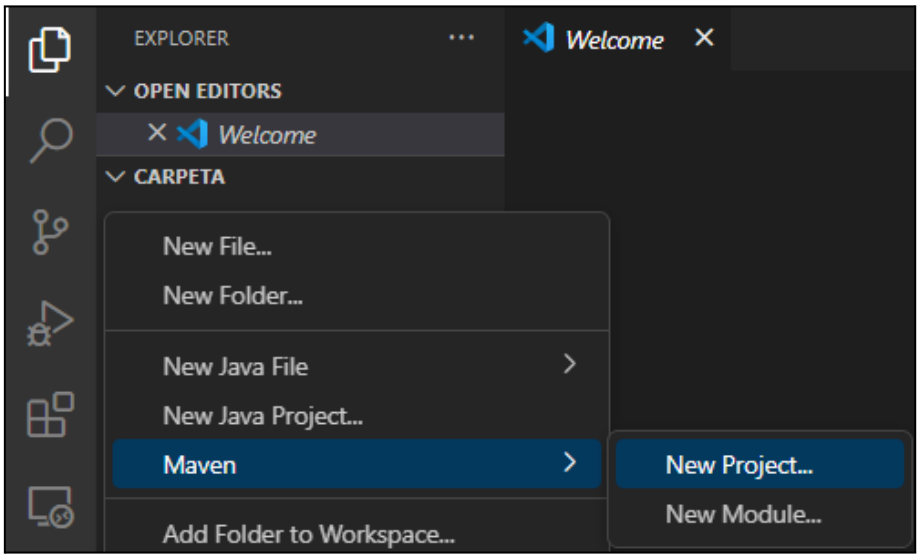
Programación con Java

TAREA 20

Alumno: Alejandro Rodríguez Jiménez
Profesor: Marc Esteve García
Curso 2024

Durante el procedimiento de esta práctica tendremos que generar algunos programas básicos en java con Swing para una interfaz gráfica y utilizando modelo-vista-controlador para la mejor gestión y visualización de nuestros archivos, por un error mio.
Para ello primero, debemos aprender cómo hacer proyectos en Maven en Visual Studio Code, empezando por descargarnos la extensión "*Maven for Java*" (esta viene dentro de otra extensión recomendada llamada "*Extension Pack for Java*").
Con esta extensión al dar click derecho en el gestor de archivos de *VSCode* tendremos la opción de "*Maven*" con la que crearemos los proyectos Maven.





Durante la documentación, para las vistas muy largas, solo mostraré los métodos en la explicación el ejercicio y el código faltante estará en "Imágenes complementarias"

Ejercicio 01:

Como primer programa deberemos crear una ventana con las operaciones básica que puede hacer, un *label* y un action listener para cambiar el tamaño del *label* a preferencia del usuario, para ello haré un *JSlider*.
Para la "*vista*" se definirá la vista y se declararán los métodos para añadir el listener y establecer el nuevo tamaño, en el "*modelo*" se guardará el tamaño para el *label* y en el "*controlador*" se creará la acción del *slider* donde se establece el tamaño.
Por último, en el main solo se harán llamadas

Vista

```java
public class Ventana extends JFrame {
    private static JLabel titulo = new JLabel(text:"Titulo prueba");
    JSlider slider;

    public Ventana() {
        setTitle(title:"Titulo interactivo");
        setSize(width:300, height:200);
        //CUANDO SE CIERRA LA VENTANA SE APAGA EL PROGRAMA
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        slider = new JSlider(min:8, max:64, value:24);

        panel.add(titulo);
        panel.add(slider);

        add(panel);
        setLocationRelativeTo(c:null);
    }

    public static void nuevoTam(Ventana ventana, int tam) {
        titulo.setFont(new Font(Font.SERIF, Font.PLAIN, tam));
        ventana.repaint();
        ventana.revalidate();
    }

    public void addSliderListener(ChangeListener listener) {
        slider.addChangeListener(listener);
    }
}
```

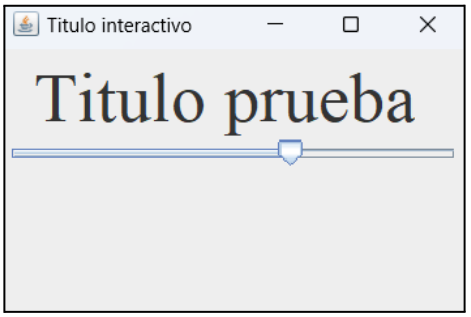
Modelo

```java
public class Modelo {
    private static int tam;

    public Modelo() {
        tam = 20;
    }

    public Modelo(int taman) {
        tam = taman;
    }

    public int getTaman() {
        return tam;
    }

    public void setTaman(int taman) {
        tam = taman;
    }
}
```

Controlador

```java
public class Controlador {
    private Ventana ventan;
    private Modelo model;

    public Controlador (Ventana ventana, Modelo modelo) {
        ventan = ventana;
        model = modelo;
        Ventana.nuevoTam(ventan, model.getTaman());

        this.ventan.addSliderListener(new SliderListener());
    }

    class SliderListener implements ChangeListener {
        @Override
        public void stateChanged(ChangeEvent e) {
            JSlider slider = (JSlider) e.getSource();
            model.setTaman(slider.getValue());
            Ventana.nuevoTam(ventan, model.getTaman());
        }
    }
}
```

Aplicación final



Ejercicio 02:

Para el segundo tendremos que hacer un programa que detecte el botón que se ha pulsado. Para hacerlo con el modelo-vista-controlador, en la "*vista*" habrán *dos labels* y los dos *botones* con las dos opciones de texto dentro de ellas, el "*modelo*" recogerá el *String* del botón al ser pulsado mediante el "*controlador*" y este se añadirá al segundo *label* de la "*vista*".

Vista

```
public class Ventana extends JFrame {
    private JLabel botonPulsado = new JLabel();
    private JButton boton1 = new JButton(text:"Judios");
    private JButton boton2 = new JButton(text:"Israelis");

    public Ventana() {
        setTitle(title:"Que botón se esta pulsando");
        setSize( width:250, height:250);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        JPanel panel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.fill = GridBagConstraints.BOTH;
        gbc.gridwidth = 2;
        gbc.weightx = 1;
        gbc.weighty = 1;
        gbc.gridy = 0;

        JLabel texto1 = new JLabel(text:"El botón pulsado és:");
        texto1.setHorizontalAlignment(SwingConstants.CENTER);
        panel.add(texto1, gbc);

        gbc.gridy = 1;
        botonPulsado.setHorizontalAlignment(SwingConstants.CENTER);
        panel.add(botonPulsado, gbc);

        gbc.gridwidth = 1;
        gbc.gridy = 2;
        panel.add(boton1, gbc);
        gbc.gridx = 1;
        panel.add(boton2, gbc);

        add(panel);
        setLocationRelativeTo(c:null);
    }

    public void setBotonPulsado(Ventana vent, String text) {
        botonPulsado.setText(text);
        vent.repaint();
        vent.revalidate();
    }

    public void addBotonListener(ActionListener listener) {
        boton1.addActionListener(listener);
        boton2.addActionListener(listener);
    }
}
```

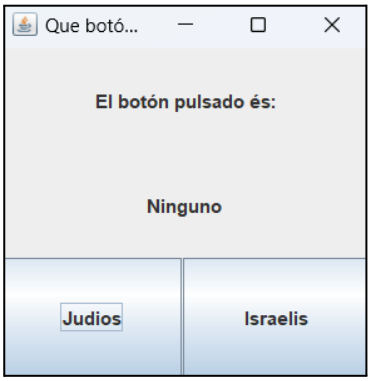
Modelo

```
public class Modelo {
    private String botonPulsado;

    public Modelo() {
        botonPulsado = "Ninguno";
    }

    public Modelo(String boton) {
        botonPulsado = boton;
    }

    public void setBotonPulsado(String pulsado) {
        botonPulsado = pulsado;
    }

    public String getBotonPulsado() {
        return botonPulsado;
    }
}
```

Controlador

```
public class Controlador {
    private Ventana ventan;
    private Modelo model;

    public Controlador (Ventana ventana, Modelo modelo) {
        ventan = ventana;
        model = modelo;
        Ventana.nuevoTam(ventan, model.getTaman());

        this.ventan.addSliderListener(new SliderListener());
    }

    class SliderListener implements ChangeListener {
        @Override
        public void stateChanged(ChangeEvent e) {
            JSlider slider = (JSlider) e.getSource();
            model.setTaman(slider.getValue());
            Ventana.nuevoTam(ventan, model.getTaman());
        }
    }
}
```

Aplicación final



Ejercicio 03:

El uso del tercer aplicativo será crear una ventana con dos botones y dos labels que almacenarán la cantidad de clicks que haga el usuario. Al utilizar el *mvc* para nuestro *modelo* almacenaremos la cantidad de clicks que se hagan en los dos botones por separado, con métodos para incrementarlos y recogerlos; en la *vista* tendremos cuatro labels y los dos botones; por último el *controlador* creará la clase para los *ActionListener* de los botones: los incrementará y recogerá su nuevo valor mostrandolo en la vista.

Métodos vista

vista

```
    public static void actualizarLabel1(Ventana vent, int nuevaCant) {
        try {
            cantClicksBot1.setText(String.valueOf(nuevaCant));
            vent.repaint();
            vent.revalidate();
        } catch (Exception e) {
            System.out.println(x:"Error indevido");
        }
    }

    public static void actualizarLabel2(Ventana vent, int nuevaCant) {
        try {
            cantClicksBot2.setText(String.valueOf(nuevaCant));
            vent.repaint();
            vent.revalidate();
        } catch (Exception e) {
            System.out.println(x:"Error indevido");
        }
    }

    public static void addBoton1Listener(ActionListener listener) {
        boton1.addActionListener(listener);
    }

    public static void addBoton2Listener(ActionListener listener) {
        boton2.addActionListener(listener);
    }
}
```

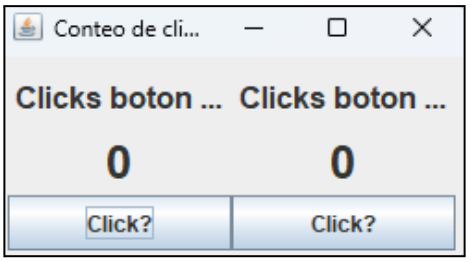
Modelo

```
public class Modelo {
    private int clicksBot1;
    private int clicksBot2;

    public Modelo() {
        clicksBot1 = 0;
        clicksBot2 = 0;
    }

    public Modelo(int click1, int click2) {
        clicksBot1 = click1;
        clicksBot2 = click2;
    }

    public void bot1Clicado() {
        clicksBot1++;
    }

    public int getClicksBot1() {
        return clicksBot1;
    }

    public void bot2Clicado() {
        clicksBot2++;
    }

    public int getClicksBot2() {
        return clicksBot2;

    }
}
```

Controlador

```
public class Controlador {
    private Ventana vent;
    private Modelo model;

    public Controlador(Ventana ventana, Modelo modelo) {
        vent = ventana;
        model = modelo;

        Ventana.addBoton1Listener(new Boton1Listener());
        Ventana.addBoton2Listener(new Boton2Listener());
    }

    class Boton1Listener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            model.bot1Clicado();
            Ventana.actualizarLabel1(vent, model.getClicksBot1());
        }
    }

    class Boton2Listener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            model.bot2Clicado();
            Ventana.actualizarLabel2(vent, model.getClicksBot2());
        }
    }
}
```

Aplicación final



Ejercicio 04:

Con el cuarto programa escucharemos las acciones que tiene una ventana en java y la mostraremos. Para nuestro *modelo* tendremos un *StringBuilder* donde podremos concatenar información y recogerla; en la *vista* tendremos un *label* de título y un *JTextArea* donde se mostrará la información y definiremos los métodos para setter la nueva información y para los *Listeners* de la pantalla; el *controlador* tendrá que crear los textos informativos para cada acción de la ventana y la agrega al *modelo* y a la *vista* con los métodos comentados.
Definiré dos *Timer* para asignar una información de la pantalla con un delay porque es una información que sino se actualiza constantemente.

Vista

```
public class Ventana extends JFrame {
    private JTextArea area = new JTextArea();

    public Ventana() {
        setTitle(title:"Eventos de la página");
        setPreferredSize(
            new Dimension(width:250, height:200));
        setMinimumSize(
            new Dimension(width:200, height:150));
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.fill = GridBagConstraints.BOTH;
        gbc.weightx = 1;
        gbc.weighty = 1;
        gbc.gridx = 0;
        gbc.gridy = 0;

        JLabel titulo = new JLabel(
            text:"- Eventos de la página -");
        titulo.setHorizontalAlignment(SwingConstants.CENTER);
        titulo.setFont(
            new Font(Font.SANS_SERIF, Font.PLAIN, size:18));
        add(titulo, gbc);

        gbc.gridy = 1;
        area.setSize(getSize());
        add(new JScrollPane(area), gbc);

        pack();
        setLocationRelativeTo(c:null);
        setVisible(b:true);
    }

    public void setArea(StringBuilder textArea) {
        area.setText(textArea.toString());
        repaint();
        revalidate();
    }

    public void addVentanaListener(WindowAdapter listener) {
        addWindowListener(listener);
    }

    public void addComponenteListener(ComponentAdapter listener) {
        addComponentListener(listener);
    }
}
```

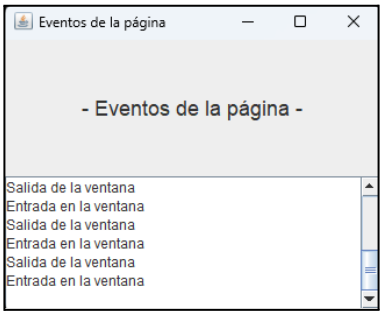
Controlador

```
public class Controlador {
    private Ventana vent; private Modelo model;
    private Timer ventanaMoviendo = new Timer(delay:500, ex -> {
        model.appendInfo(texto:"Moviendo ventana...");
        vent.setArea(model.getTexto());    });
    private Timer ventanaReescalando = new Timer(delay:500, ex -> {
        model.appendInfo(texto:"Reescalando ventana...");
        vent.setArea(model.getTexto());    });

    public Controlador(Ventana ventana, Modelo modelo) {
        vent = ventana; model = modelo;

        vent.addVentanaListener(new AccionesListener());
        vent.addComponenteListener(new ComponentesListener());
    }

    class AccionesListener extends WindowAdapter {
        @Override
        public void windowOpened(WindowEvent e) {
            model.appendInfo(texto:"Ventana Abierta");
            vent.setArea(model.getTexto());    }
        @Override
        public void windowIconified(WindowEvent e) {
            model.appendInfo(texto:"Ventana Minimizada");
            vent.setArea(model.getTexto());    }
        @Override
        public void windowActivated(WindowEvent e) {
            model.appendInfo(texto:"Entrada en la ventana");
            vent.setArea(model.getTexto());    }
        @Override
        public void windowDeactivated(WindowEvent e) {
            model.appendInfo(texto:"Salida de la ventana");
            vent.setArea(model.getTexto());    }
    }

    class ComponentesListener extends ComponentAdapter {
        Point initialLocation;
        @Override
        public void componentResized(ComponentEvent e) {
            if (vent.getExtendedState() == JFrame.MAXIMIZED_BOTH) {
                model.appendInfo(texto:"Ventana Maximizada");
                vent.setArea(model.getTexto());
            } else if (vent.getExtendedState() == JFrame.NORMAL) {
                ventanaReescalando.setRepeats(flag:false);
                ventanaReescalando.start();    }    }
        @Override
        public void componentMoved(ComponentEvent e) {
            if (initialLocation == null) {
                initialLocation = vent.getLocation();
            } else {
                Point currentLocation = vent.getLocation();
                if (!currentLocation.equals(initialLocation)) {
                    ventanaMoviendo.setRepeats(flag:false);
                    ventanaMoviendo.start();    }    }    }
    }
}
```

Modelo

```
public class Modelo {
    private StringBuilder area = new StringBuilder();

    public Modelo() {
    }

    public StringBuilder getTexto() {
        return area;
    }

    public void appendInfo(String texto) {
        area.append(texto + "\n");
    }
}
```

Aplicación final



Ejercicio 05:

La quinta aplicación será parecida a la anterior, en vez de recoger las entradas de la ventana las recogeremos del ratón, y de igual manera lo mostraremos en pantalla. Pero tendremos que añadir un botón con el que podremos limpiar el cuadro de texto de las acciones del ratón.

Vista

```java
public class Ventana extends JFrame{
    private JButton limpiar = new JButton(text:"L I M P I A R");
    private JTextArea infoRaton = new JTextArea();

    public Ventana() {
        setTitle(title:"Acciones del ratón");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setPreferredSize(new Dimension(width:200, height:200));

        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;

        JLabel titulo = new JLabel(text:"- Eventos del ratón -");
        titulo.setHorizontalTextPosition(SwingConstants.CENTER);
        titulo.setFont(new Font(Font.SANS_SERIF, Font.PLAIN, size:18));
        add(titulo, gbc);

        gbc.gridy = 1;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        limpiar.setAlignmentX(Component.CENTER_ALIGNMENT);
        add(limpiar, gbc);

        gbc.gridy = 2;
        infoRaton.setEditable(b:false);
        infoRaton.setPreferredSize(new Dimension(width:100, height:100));
        add(new JScrollPane(infoRaton), gbc);

        pack();
        setLocationRelativeTo(c:null);
        setVisible(b:true);
    }

    public void setInformacionRaton(StringBuilder nuevoTexto) {
        infoRaton.setText(nuevoTexto.toString());
        repaint();
        revalidate();
        pack();
    }

    public void addLimpiarListener(ActionListener listener) {
        limpiar.addActionListener(listener);
    }

    public void addRatonListener(MouseAdapter listener) {
        addMouseListener(listener);
    }
}
```

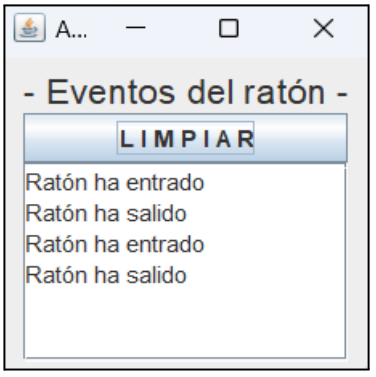
Controlador

```java
public class Controlador {
    Ventana ventan;
    Modelo model;
    Cursor cursor;

    public Controlador(Ventana ventana, Modelo modelo) {
        ventan = ventana;
        model = modelo;

        ventan.setInformacionRaton(model.getTexto());
        ventan.addMouseListener(new miMouseListener());
        ventan.addLimpiarListener(new LimpiarListener());
    }

    class LimpiarListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            model.setTexto(texto:"");
            ventan.setInformacionRaton(model.getTexto());
        }
    }

    class miMouseListener extends MouseAdapter {
        @Override
        public void mouseClicked(MouseEvent e) {
            if (e.getButton() == MouseEvent.BUTTON1) {
                model.appendInfo(texto:"Click izquierdo");
            } else if (e.getButton() == MouseEvent.BUTTON2) {
                model.appendInfo(texto:"Click medio");
            } else {
                model.appendInfo(texto:"Click derecho");
            }
            ventan.setInformacionRaton(model.getTexto());
        }
        @Override
        public void mouseEntered(java.awt.event.MouseEvent e) {
            model.appendInfo(texto:"Ratón ha entrado");
            ventan.setInformacionRaton(model.getTexto());
        }
        @Override
        public void mouseExited(java.awt.event.MouseEvent e) {
            model.appendInfo(texto:"Ratón ha salido");
            ventan.setInformacionRaton(model.getTexto());
        }
    };

    class addMouseWheelListener implements MouseWheelListener {
        @Override
        public void mouseWheelMoved(MouseWheelEvent e) {
            if (e.getWheelRotation() < 0) {
                model.appendInfo(texto:"Rueda hacia arriba");
            } else {
                model.appendInfo(texto:"Rueda hacia abajo");
            }
            ventan.setInformacionRaton(model.getTexto());
        }
    };
}
```

Modelo

```java
public class Modelo {
    private StringBuilder accionesRaton = new StringBuilder();

    public Modelo() {
    }

    public StringBuilder getTexto() {
        return accionesRaton;
    }

    public void setTexto(String texto) {
        accionesRaton = new StringBuilder(texto);
    }

    public void appendInfo(String texto) {
        accionesRaton.append(texto + "\n");
    }
}
```

Aplicación final



Ejercicio 06:

Con nuestra sexta aplicación gráfica podremos calcular el IMC del usuario al pasarnos la altura en metros y el peso en kilogramos. Para ello en nuestra *vista* pondremos dos *JTextField*, un *JTextArea* y un *JButton*, añadiremos sus métodos getters para los *JTextField*, el método setter para el *JTextArea* y el método de acción del *JButton*; con nuestro *modelo* recogeremos los valores de altura y peso para calcular el IMC, y definiremos los métodos getters & setters para todas las variables; por último, en el *controlador* crearemos el *class* donde estableceremos las variables de altura y peso en el *modelo* y recogeremos el IMC y lo estableceremos en la *vista*.

Modelo

```java
public class Modelo {
    private double mAltura;
    private double peso;
    private double imc;

    public Modelo() {
    }

    public Modelo(int cmAlt, double pes) {
        mAltura = cmAlt / 100.0;
        peso = pes;
        double elCuadrado = Math.pow(mAltura, b:2);
        imc = peso / elCuadrado;
    }

    public double getMetrosAltura() {
        return mAltura;
    }

    public void setMetrosAltura(double metros) {
        mAltura = metros;
    }

    public double getPeso() {
        return peso;
    }

    public void setPeso(double pes) {
        peso = pes;
    }

    public double getIMC() {
        return imc;
    }

    public void setIMC(double im) {
        imc = im;
    }
}
```

Métodos vista

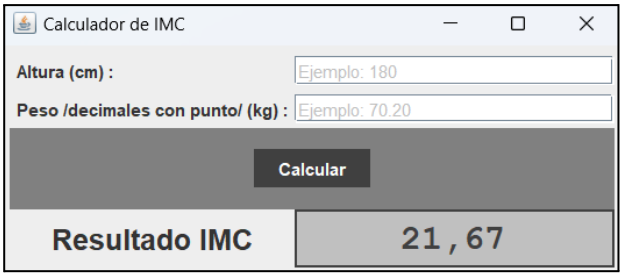
vista

```java
public void textoQueDesaparece(JTextField textField, String texto) {
    final Object[] objetos = {textField, texto};
    textField.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseEntered(MouseEvent e) {
            if (((JTextField) objetos[0]).getText().equals(((String) objetos[1]))) {
                ((JTextField) objetos[0]).setText(t:"");
                ((JTextField) objetos[0]).setForeground(Color.BLACK);
            }
        }

        @Override
        public void mouseExited(MouseEvent e) {
            if (((JTextField) objetos[0]).getText().equals(anObject:"")) {
                ((JTextField) objetos[0]).setText(((String) objetos[1]));
                ((JTextField) objetos[0]).setForeground(Color.LIGHT_GRAY);
            }
        }
    });
}

public static int getAltura() {
    return Integer.parseInt(altura.getText());
}

public static double getPeso() {
    return Double.parseDouble(peso.getText());
}

public static void setImc(Ventana vent, double imcIntroducido) {
    imc.setText(String.format(format:"%.2f", imcIntroducido));
    vent.repaint();
    vent.revalidate();
}

public void addCalcularListener(ActionListener listener) {
    calcular.addActionListener(listener);
}
```

Controlador

```java
public class Controlador {
    private Ventana vent;
    private Modelo model;

    public Controlador(Ventana ventana, Modelo modelo) {
        vent = ventana;
        model = modelo;

        vent.addCalcularListener(new CalcularListener());
    }

    class CalcularListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                int altura = Ventana.getAltura();
                double peso = Ventana.getPeso();
                model = new Modelo(altura, peso);
                double imc = model.getIMC();
                Ventana.setImc(vent, imc);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(vent,
                "Por favor, ingrese valores "
                + "válidos para altura y peso.",
                title:"Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}
```

Aplicación final



Ejercicio 07:

En el séptimo programa generaremos un conversor de divisas de pesetas a euros. La *vista* contendrá un *JTextField*, dinero ingresado; un *JTextPane,* conversión; y dos *JButton*, invertir el sentido de conversión y convertir; en el *modelo* recogeremos el dinero a convertir y el sentido de conversión, y lo convertirá a la divisa contraria; por último, el *controlador* recogerá el estado y el dinero a convertir y lo pasará al *modelo* para recoger la conversión y pasarla a la *vista*.

Métodos vista

```
public void setEstadoConvertor (boolean estadoConversor) {
    euros = estadoConversor;
}

public boolean getEstadoConvertor () {
    return euros;
}

public void setTextoBotonCambiante(String texto) {
    cambiarValor.setText(texto);
}

public double getDineroAConvertir() {
    double d = 0;
    try {
        d = Double.parseDouble(dineroSinConvertir.getText());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(parentComponent:null,message:"Valor inválido",
         "Error " + e.getCause(), JOptionPane.ERROR_MESSAGE);
    }
    return d;
}
```

```
public void textoQueCambia(Ventana vent) {
    String resultado = (euros) ? "Dinero en euros : " : "Dinero en pesetas : ";
    textoDinCon.setText(resultado);
    repaint();
    revalidate();
}

public static void setDineroConvertido(Ventana vent, double dinConvertido) {
    String texto = "";
    if (euros) {
        texto = String.format(format:"%.3f", dinConvertido);
    } else {
        texto = String.format(format:"%.2f", dinConvertido);
    }
    dineroConvertido.setText(texto);
    vent.repaint();
    vent.revalidate();
}

public void addCambiarValorListener(ActionListener listener) {
    cambiarValor.addActionListener(listener);
}

public void addConvertirListener(ActionListener listener) {
    convertir.addActionListener(listener);
}
```

Modelo

```
public class Modelo {
    private double dineroEntrada;
    private double dineroSalida;
    private boolean euros;

    public Modelo() {
        dineroEntrada = 0;
        dineroSalida = 0;
        euros = true;
    }

    public double getDineroEntrada() {
        return dineroEntrada;
    }

    public double getDineroSalida() {
        return dineroSalida;
    }
```

```
public Modelo(double dineroEn, boolean estadoActual) {
    dineroEntrada = dineroEn;
    euros = estadoActual;
    if (euros) {
        dineroSalida = dineroEn * 166.386;
    } else {
        dineroSalida = dineroEn * 0.006;
    }
}

public void setBooleanEuros(boolean estadoConversor) {
    euros = estadoConversor;
}
```

Controlador

```
public class Controlador {
    Ventana vent;
    Modelo model;

    public Controlador(Ventana ventana, Modelo modelo) {
        vent = ventana;
        model = modelo;

        vent.addCambiarValorListener(new CambiarValorListener());
        vent.addConvertirListener(new ConvertirListener());
    }

    class CambiarValorListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            boolean estado = (vent.getEstadoConvertor()) ? false : true;
            model.setBooleanEuros(estado);
            vent.setEstadoConvertor(estado);
            String divisa = (estado) ? "CONV. A PESETAS" : "CONV. A EUROS";
            vent.setTextoBotonCambiante(divisa);
            vent.textoQueCambia(vent);
        }
    }

    class ConvertirListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            double dinero = vent.getDineroAConvertir();
            model = new Modelo(dinero, vent.getEstadoConvertor());
            Ventana.setDineroConvertido(vent, model.getDineroSalida());
        }
    }
}
```

Aplicación final

Ejercicio 08:

Para el octavo aplicativo mejoraremos el aplicativo del ejercicio anterior añadiendo un botón para eliminar los datos, la posibilidad de usar los botones del teclado y manejando la posibilidad de errores. Como es prácticamente lo mismo me centraré en estos casos dejando capturas de todo el código al final en "Imágenes ejercicio08".

Clases del controlador para el nuevo botón y las teclas

```
class LimpiarListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        vent.setDineroParaConvertir(d:0.0);
        vent.setDineroConvertido(dinConvertido:0.0);
    }
}

class TecladoListener extends KeyAdapter {
    @Override
    public void keyTyped(KeyEvent e) {
        char keyChar = e.getKeyChar();

        if (Character.isDigit(keyChar)) {
            JTextField tf = vent.getTextFieldDinero();
            tf.setForeground(Color.BLACK);
            double num = Double.parseDouble(tf.getText());
            String texto = (num > 0) ?  tf.getText() + keyChar : "" + keyChar;
            vent.setDineroParaConvertir(Double.parseDouble(texto.toString()));
        } else if (keyChar == KeyEvent.VK_ENTER) {
            double dinero = vent.getDineroAConvertir();
            model = new Modelo(dinero, vent.getEstadoConvertor());
            vent.setDineroConvertido(model.getDineroSalida());
        } else {
            e.consume();
        }
    }
}
```

Ejercicio 09:

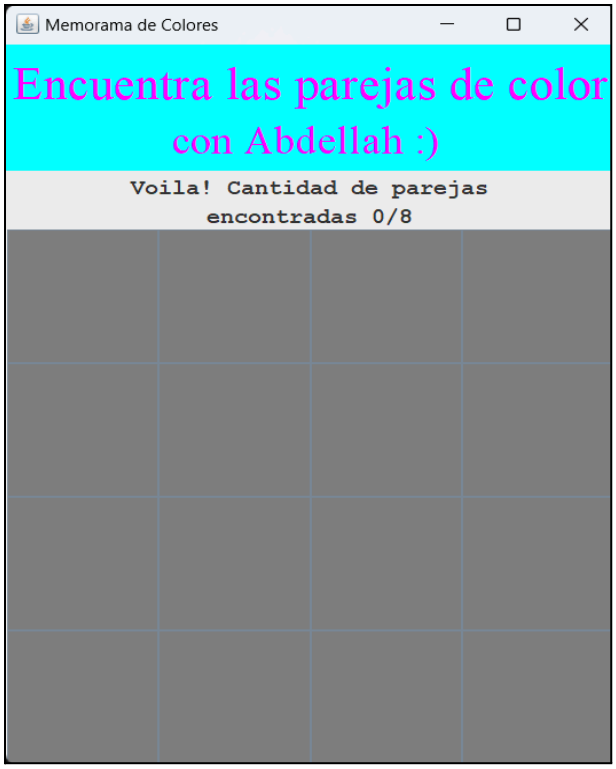
Para el último programa crearemos el juego *"Memory"* en grupos. Para ello, crearemos una vista con una tabla 4 x 4 donde colocaremos los botones con distintos colores, la gran parte de lógica se encontrará al pulsar el segundo botón donde: primero desactiva la posibilidad de pulsar más botones mientras se comprueban los colores, al no ser iguales volverá a "taparlas" ocultando sus colores en cambio si son iguales, se quedarán "descubiertas" y se añadirá uno al contador de parejas. A la hora de completar el juego y encontrar a todas las parejas saldrá un mensaje donde nos felicitará; y s por el contrario cometemos varios errores, en concreto 5, se reiniciará el nivel "tapando todos los botones".

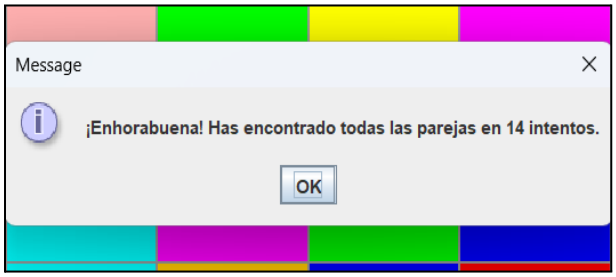
Vista

```
public class App extends JFrame {
    private JButton[] buttons = new JButton[16];
    private ArrayList<Color> colors = new ArrayList<>();
    private JButton firstButton = null;
    private JButton secondButton = null;
    private Color firstColor;
    private Color secondColor;
    private boolean isComparing = false;
    private int pairsFound = 0;
    private JPanel panelParejas = new JPanel();
    private JPanel panelBotones = new JPanel();
    private StringBuilder TotalParejas = new StringBuilder ();
    private int intentos = 0;
    private int intentosTotales = 0;

    public App() {
        setTitle(title:"Memorama de Colores");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridBagLayout());
        GridBagConstraints pos = new GridBagConstraints();
        pos.fill = GridBagConstraints.BOTH;
        pos.gridx = 0;

        Color[] colorArray = {
            Color.RED, Color.BLUE, Color.GREEN, Color.YELLOW,
            Color.ORANGE, Color.CYAN, Color.MAGENTA, Color.PINK
        };

        for (Color color : colorArray) {
            colors.add(color);
            colors.add(color);
        }
        Collections.shuffle(colors);
```

```
JPanel panel = new JPanel();
panel.setBorder(new EmptyBorder(top:5,left:5,bottom:5,right:5));
panel.setBackground(Color.CYAN);
panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
JLabel titul1 = new JLabel(text:"Encuentra las parejas de color");
JLabel titul2 = new JLabel(text:"con Abdellah :) ");
titul1.setFont(new Font(Font.SERIF, Font.PLAIN, size:35));
titul2.setFont(new Font(Font.SERIF, Font.PLAIN, size:30));
titul1.setForeground(Color.MAGENTA);
titul2.setForeground(Color.MAGENTA);
titul1.setAlignmentX(Component.CENTER_ALIGNMENT);
titul2.setAlignmentX(Component.CENTER_ALIGNMENT);
panel.add(titul1);
panel.add(titul2);
pos.gridy = 0;
add(panel, pos);

ParejasTotales(TotalParejas, pairsFound);
pos.gridy = 1;
add(panelParejas, pos);

panelBotones.setPreferredSize(new Dimension(width:400, height:400));
panelBotones.setLayout(new GridLayout(rows:4, cols:4));

for (int i = 0; i < 16; i++) {
    buttons[i] = new JButton();
    buttons[i].setBackground(Color.GRAY);
    buttons[i].addActionListener(new ButtonClickListener());
    panelBotones.add(buttons[i]);
}
pos.gridy = 2;
add(panelBotones, pos);
pack();
setLocationRelativeTo(c:null);    }
```

```
private class ButtonClickListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (isComparing) {
            return;
        }
        JButton clickedButton = (JButton) e.getSource();
        int index = indiceBoton(clickedButton);
        if (clickedButton.getBackground().equals(Color.GRAY) && firstButton != clickedButton) {
            clickedButton.setBackground(colors.get(index));
            if (firstButton == null) {
                firstButton = clickedButton;
                firstColor = colors.get(index);
            } else {
                intentos++;
                intentosTotales++;
                secondButton = clickedButton;
                secondColor = colors.get(index);
                isComparing = true;
                if (firstColor.equals(secondColor)) {
                    intentos = 0;
                    firstButton.setEnabled(b:false);
                    secondButton.setEnabled(b:false);
                    pairsFound++;
                    panelParejas.removeAll();
                    ParejasTotales(TotalParejas, pairsFound);
                    panelParejas.revalidate();
                    panelParejas.repaint();
                    resetSelection();
                    if (pairsFound == 8) {
                        JOptionPane.showMessageDialog(parentComponent:null, "¡Enhorabuena! Has"
                        + " encontrado todas las parejas en " + intentosTotales + " intentos.");
                        intentosTotales = 0;    }
                } else {
                    Timer timer = new Timer(delay:500, new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            firstButton.setBackground(Color.GRAY);
                            secondButton.setBackground(Color.GRAY);    }    });
                    timer.setRepeats(flag:false);
                    timer.start();    }    }
                if(intentos >= 5) {
                    intentos();
                    pairsFound = 0;
                    panelParejas.removeAll();
                    ParejasTotales(TotalParejas, pairsFound);
                    panelParejas.revalidate();
                    panelParejas.repaint();    }    }    }    }
```






```java
private void resetSelection() {
    firstButton = null; secondButton = null; isComparing = false;
}

public void ParejasTotales (StringBuilder TotalParejas, int pairsFound) {
    panelParejas.setLayout(new BoxLayout(panelParejas, BoxLayout.Y_AXIS));
    TotalParejas = new StringBuilder (str:" Voila! Cantidad de parejas &encontradas ");
    TotalParejas.append(pairsFound).append(str:"/8");
    String[] textos = TotalParejas.toString().split(regex:"&");
    JLabel parejasCont1 = new JLabel(textos[0]);
    JLabel parejasCont2 = new JLabel(textos[1]);
    parejasCont1.setFont(new Font(Font.MONOSPACED, Font.BOLD, size:16));
    parejasCont2.setFont(new Font(Font.MONOSPACED, Font.BOLD, size:16));
    parejasCont1.setAlignmentX(Component.CENTER_ALIGNMENT);
    parejasCont2.setAlignmentX(Component.CENTER_ALIGNMENT);

    panelParejas.add(parejasCont1);
    panelParejas.add(parejasCont2);
}

public int indiceBoton(JButton clickedButton) {
    int index = 0;
    for (int i = 0; i < buttons.length; i++) {
        if (clickedButton == buttons[i]) {
            index = i;
            break;
        }
    }
    return index;
}

public void intentos () {
    for (Object o : panelBotones.getComponents()) {
        if (o instanceof JButton) {
            JButton Button = (JButton)o;
            Button.setBackground(Color.gray);
            Button.setEnabled(b:true);
        }
    }
    panelBotones.repaint();
    panelBotones.revalidate();
    intentos = 0;
}
```

Vista ejercicio03:

```
public class Ventana extends JFrame {
    private static JLabel cantClicksBot1;
    private static JLabel cantClicksBot2;
    private static JButton boton1 = new JButton(text:"Click?");
    private static JButton boton2 = new JButton(text:"Click?");

    public Ventana() {
        setTitle(title:"Conteo de clicks");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setMinimumSize(new Dimension(width:0, height:150));
        EmptyBorder borde = new EmptyBorder(top:4, left:4, bottom:4, right:4);

        JPanel panel = new JPanel(new GridBagLayout());
        panel.setBorder(new EmptyBorder(top:5, left:5, bottom:5, right:5));
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.fill = GridBagConstraints.BOTH;
        gbc.weightx = 1;
        gbc.weighty = 1;
        gbc.gridy = 0;
        gbc.gridx = 0;

        JLabel labelBot1 = new JLabel(text:"Clicks boton 1:");
        Font nuevaFuente1 = labelBot1.getFont().deriveFont(size:16f);
        labelBot1.setFont(nuevaFuente1);
        labelBot1.setBorder(borde);
        panel.add(labelBot1, gbc);

        gbc.gridx = 1;
        JLabel labelBot2 = new JLabel(text:"Clicks boton 2:");
        Font nuevaFuente2 = labelBot2.getFont().deriveFont(size:16f);
        labelBot2.setFont(nuevaFuente2);
        labelBot2.setBorder(borde);
        panel.add(labelBot2, gbc);
```

```
        gbc.gridx = 1;
        cantClicksBot2 = new JLabel(text:"0");
        cantClicksBot2.setFont(new Font(Font.SANS_SERIF, Font.BOLD, size:25));
        cantClicksBot2.setHorizontalAlignment(SwingConstants.CENTER);
        panel.add(cantClicksBot2, gbc);

        gbc.gridy = 2;
        gbc.gridx = 0;
        panel.add(boton1, gbc);

        gbc.gridx = 1;
        panel.add(boton2, gbc);

        add(panel);

        pack();
        setLocationRelativeTo(c:null);
    }
```

Vista ejercicio06:

```
public class Ventana extends JFrame{
    private static JButton calcular = new JButton(text:"Calcular");
    private static JTextField altura = new JTextField(columns:20);
    private static JTextField peso = new JTextField(columns:20);
    private static JTextPane imc = new JTextPane();

    public Ventana() {
        setTitle(title:"Calculador de IMC");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(width:300, height:250);

        EmptyBorder borde = new EmptyBorder(top:5, left:5, bottom:5, right:5);
        JPanel panel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridwidth = 1;
        gbc.gridy = 0;
        gbc.gridx = 0;

        JLabel meteAltura = new JLabel(text:"Altura (cm) :");
        meteAltura.setBorder(borde);
        panel.add(meteAltura, gbc);

        gbc.gridx = 1;
        altura.setText(t:"Ejemplo: 180");
        altura.setForeground(Color.LIGHT_GRAY);
        textoQueDesaparece(altura, texto:"Ejemplo: 180");
        panel.add(altura, gbc);

        gbc.gridy = 1;
        gbc.gridx = 0;
        JLabel metePeso = new JLabel(text:"Peso /decimales con punto/ (kg) :");
        metePeso.setBorder(borde);
        panel.add(metePeso, gbc);

        gbc.gridx = 1;
        peso.setText(t:"Ejemplo: 70.20");
        peso.setForeground(Color.LIGHT_GRAY);
        textoQueDesaparece(peso, texto:"Ejemplo: 70.20");
        panel.add(peso, gbc);
```

```
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridwidth = 2;
        gbc.gridy = 2;
        gbc.gridx = 0;
        JPanel panelBoton = new JPanel();
        panelBoton.setBackground(Color.GRAY);
        panelBoton.setBorder(new EmptyBorder(top:10, left:0, bottom:10, right:0));
        calcular.setBackground(Color.DARK_GRAY);
        calcular.setForeground(Color.WHITE);
        calcular.setBorderPainted(b:false);
        panelBoton.add(calcular);
        panel.add(panelBoton, gbc);

        gbc.fill = GridBagConstraints.NONE;
        gbc.gridwidth = 1;
        gbc.gridy = 3;
        gbc.gridx = 0;
        JLabel calcularLabel = new JLabel(text:"Resultado IMC");
        calcularLabel.setFont(new Font(Font.SANS_SERIF, Font.BOLD, size:20));
        calcularLabel.setBorder(borde);
        panel.add(calcularLabel, gbc);

        gbc.gridx = 1;
        StyledDocument doc = imc.getStyledDocument();
        SimpleAttributeSet center = new SimpleAttributeSet();
        StyleConstants.setAlignment(center, StyleConstants.ALIGN_CENTER);
        doc.setParagraphAttributes(offset:0, doc.getLength(), center, replace:false);

        imc.setBorder(new LineBorder(Color.DARK_GRAY, thickness:2));
        imc.setBackground(Color.LIGHT_GRAY);
        imc.setEditable(b:false);
        imc.setFont(new Font(Font.MONOSPACED, Font.BOLD, size:24));
        imc.setForeground(Color.DARK_GRAY);
        imc.setPreferredSize(new Dimension(width:220, height:40));
        imc.setText(t:"10");
        panel.add(imc, gbc);

        panel.setBorder(borde);
        add(panel);

        setImc(this, imcIntroducido:21.67);

        pack();
        setLocationRelativeTo(c:null);
        setVisible(b:true);
    }
```

Ejercicio08:

Modelo

```java
public class Modelo {
    private double dineroEntrada;
    private double dineroSalida;
    private boolean euros;
```

```java
public Modelo(double dineroEn, boolean estadoActual) {
    dineroEntrada = dineroEn;
    euros = estadoActual;
    if (euros) {
        dineroSalida = dineroEn * 166.386;
    } else {
        dineroSalida = dineroEn * 0.006;
    }
}
```

```java
public void setBooleanEuros(boolean estadoConversor) {
    euros = estadoConversor;
}

public double getDineroEntrada() {
    return dineroEntrada;
}

public double getDineroSalida() {
    return dineroSalida;
}
```

Controlador

```java
public class Controlador {
    Ventana vent;
    Modelo model;

    public Controlador(Ventana ventana, Modelo modelo) {
        vent = ventana;
        model = modelo;

        vent.addListeners(new LimpiarListener(), new CambiarValorListener(),
        new ConvertirListener(), new TecladoListener());
    }

    class CambiarValorListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            boolean estado = (vent.getEstadoConvertor()) ? false : true;
            model.setBooleanEuros(estado);
            vent.setEstadoConvertor(estado);
            String divisa = (estado) ? "CONV. A PESETAS" : "CONV. A EUROS";
            vent.textoQueCambia(vent, divisa);
        }
    }

    class ConvertirListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            double dinero = vent.getDineroAConvertir();
            model = new Modelo(dinero, vent.getEstadoConvertor());
            vent.setDineroConvertido(model.getDineroSalida());
        }
    }
```

```java
class LimpiarListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        vent.setDineroParaConvertir(d:0.0);
        vent.setDineroConvertido(dinConvertido:0.0);
    }
}

class TecladoListener extends KeyAdapter {
    @Override
    public void keyTyped(KeyEvent e) {
        char keyChar = e.getKeyChar();

        if (Character.isDigit(keyChar)) {
            JTextField tf = vent.getTextFieldDinero();
            tf.setForeground(Color.BLACK);
            double num = Double.parseDouble(tf.getText());
            String texto = (num > 0) ?  tf.getText() + keyChar : "" + keyChar;
            vent.setDineroParaConvertir(Double.parseDouble(texto.toString()));
        } else if (keyChar == KeyEvent.VK_ENTER) {
            double dinero = vent.getDineroAConvertir();
            model = new Modelo(dinero, vent.getEstadoConvertor());
            vent.setDineroConvertido(model.getDineroSalida());
        } else {
            e.consume();
        }
    }
}
```

Métodos vista

```java
public void setEstadoConvertor (boolean estadoConversor) {
    euros = estadoConversor;
}

public boolean getEstadoConvertor () {
    return euros;
}

public JTextField getTextFieldDinero() {
    return dineroSinConvertir;
}

public void setDineroParaConvertir(double d) {
    dineroSinConvertir.setText(String.valueOf(d));
}

public void setTextoBotonCambiante(String texto) {
    cambiarValor.setText(texto);
}
```

```java
public void caracBotones() {
    cambiarValor.setBorderPainted(b:false);
    cambiarValor.setBackground(Color.DARK_GRAY);
    cambiarValor.setForeground(Color.WHITE);
    cambiarValor.setAlignmentX(CENTER_ALIGNMENT);

    convertir.setBorderPainted(b:false);
    convertir.setBackground(Color.DARK_GRAY);
    convertir.setForeground(Color.WHITE);
    convertir.setAlignmentX(CENTER_ALIGNMENT);
}
```

Vista

```
public Ventana() {
    euros = true;
    setTitle(title:"Cenversor euros - pesetas / pesetas - euros");
    setDefaultCloseOperation(EXIT_ON_CLOSE);

    JPanel panel = new JPanel();
    panel.setLayout(new GridBagLayout());
    JLabel ingresaDinero = new JLabel(text:"Ingrese la cantidad a convertir : ");
    JPanel panelTF = new JPanel();
    panelTF.add(dineroSinConvertir);
    JPanel panelTP = new JPanel();
    panelTP.add(dineroConvertido);
    JPanel panelBotones = new JPanel(new GridLayout(rows:1, cols:2));
    panelBotones.add(cambiarValor);
    panelBotones.add(convertir);
    addCaracteristicas(panelTF, panelTP, panelBotones);

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(top:5, left:5, bottom:0, right:5);
    gbc.fill = GridBagConstraints.BOTH;
    gbc.gridwidth = 1; gbc.gridy = 0; gbc.gridx = 0;
    panel.add(ingresaDinero, gbc);

    gbc.gridx = 1;
    panel.add(panelTF, gbc);

    gbc.insets = new Insets(top:0, left:5, bottom:0, right:5);
    gbc.gridy = 1; gbc.gridx = 0;
    panel.add(textoDinCon, gbc);

    gbc.gridx = 1;
    panel.add(panelTP, gbc);

    gbc.insets = new Insets(top:0, left:5, bottom:2, right:5);
    gbc.gridy = 2; gbc.gridx = 0; gbc.gridwidth = 2;
    panel.add(panelBotones, gbc);

    gbc.insets = new Insets(top:1, left:5, bottom:5, right:5);
    gbc.fill = GridBagConstraints.HORIZONTAL; gbc.gridy = 3; gbc.gridx = 0;
    panel.add(limpiar, gbc);

    add(panel);

    pack();
    setLocationRelativeTo(c:null);
    setVisible(b:true);
}
```

```
public void addCaracteristicas(JPanel panelTF, JPanel panelTP, JPanel panelBotones) {
    String resultado = (euros) ? "Dinero en euros : " : "Dinero en pesetas : ";
    textoDinCon.setText(resultado);
    textoDinCon.setBorder(new EmptyBorder(top:0, left:0, bottom:5, right:0));

    panelTF.setBorder(new LineBorder(Color.BLACK, thickness:1));
    panelTF.setBackground(Color.GRAY);
    dineroSinConvertir.setBorder(new EmptyBorder(top:5, left:5, bottom:5, right:5));
    dineroSinConvertir.setBackground(Color.GRAY);
    dineroSinConvertir.setHorizontalAlignment(JTextField.CENTER);
    dineroSinConvertir.setText(t:"1");
    dineroSinConvertir.setForeground(Color.LIGHT_GRAY);
    textoQueDesaparece(dineroSinConvertir, texto:"1");

    StyledDocument doc = dineroConvertido.getStyledDocument();
    SimpleAttributeSet center = new SimpleAttributeSet();
    StyleConstants.setAlignment(center, StyleConstants.ALIGN_CENTER);
    doc.setParagraphAttributes(offset:0, doc.getLength(), center, replace:false);

    panelTP.setBorder(new LineBorder(Color.BLACK, thickness:1));
    panelTP.setBackground(Color.LIGHT_GRAY);
    dineroConvertido.setEditable(b:false);
    dineroConvertido.setPreferredSize(dineroSinConvertir.getPreferredSize());
    dineroConvertido.setBackground(Color.LIGHT_GRAY);
    dineroConvertido.setForeground(Color.GRAY);
    dineroConvertido.setText(t:"166.386");

    panelBotones.setBackground(Color.GRAY);

    cambiarValor.setBorderPainted(b:false);
    cambiarValor.setBackground(Color.DARK_GRAY);
    cambiarValor.setForeground(Color.WHITE);
    cambiarValor.setAlignmentX(CENTER_ALIGNMENT);

    convertir.setBorderPainted(b:false);
    convertir.setBackground(Color.DARK_GRAY);
    convertir.setForeground(Color.WHITE);
    convertir.setAlignmentX(CENTER_ALIGNMENT);
}
```

```java
public void textoQueDesaparece(JTextField textField, String texto) {
    final Object[] objetos = {textField, texto};
    textField.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseEntered(MouseEvent e) {
            if (((JTextField) objetos[0]).getText().equals(((String) objetos[1])) ||
            ((JTextField) objetos[0]).getText().equals(anObject:"0.0")) {
                ((JTextField) objetos[0]).setText(t:"");
                ((JTextField) objetos[0]).setForeground(Color.BLACK);    }    }
        @Override
        public void mouseExited(MouseEvent e) {
            if (((JTextField) objetos[0]).getText().equals(anObject:"")) {
                ((JTextField) objetos[0]).setText(((String) objetos[1]));
                ((JTextField) objetos[0]).setForeground(Color.LIGHT_GRAY);    }    }
    });
}

public void textoQueCambia(Ventana vent, String texto) {
    String resultado = (euros) ? "Dinero en euros : " : "Dinero en pesetas : ";
    cambiarValor.setText(texto);
    textoDinCon.setText(resultado);
    repaint();
    revalidate();
}
```

```java
public double getDineroAConvertir() {
    double d = 0;
    try {
        d = Double.parseDouble(dineroSinConvertir.getText());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"Valor inválido",
            "Error " + e.getCause(), JOptionPane.ERROR_MESSAGE);
    }
    return d;
}

public void setDineroConvertido(double dinConvertido) {
    String texto = "";
    if (euros) {
        texto = String.format(format:"%.3f", dinConvertido);
    } else {
        texto = String.format(format:"%.2f", dinConvertido);
    }
    dineroConvertido.setText(texto);
    repaint();
    revalidate();
}
```

```java
public boolean getEstadoConvertor () {
    return euros;
}

public void setEstadoConvertor (boolean estadoConversor) {
    euros = estadoConversor;
}

public JTextField getTextFieldDinero() {
    return dineroSinConvertir;
}

public void setDineroParaConvertir(double d) {
    dineroSinConvertir.setText(String.valueOf(d));
}

public void addListeners(ActionListener listenerLimpiar,
ActionListener listenerCambiarValor,
ActionListener listenerConvertir, KeyListener listenerTeclado) {
    limpiar.addActionListener(listenerLimpiar);
    cambiarValor.addActionListener(listenerCambiarValor);
    convertir.addActionListener(listenerConvertir);
    dineroSinConvertir.addKeyListener(listenerTeclado);
}
```

```java
public class Ventana extends JFrame {
    private JButton limpiar = new JButton(text:"LIMPIAR");
    private JButton cambiarValor = new JButton(text:"CONV. A PESETAS");
    private JButton convertir = new JButton(text:"CONVERTIR");
    private JTextField dineroSinConvertir = new JTextField(columns:20);
    private JTextPane dineroConvertido = new JTextPane();
    private boolean euros;
    private JLabel textoDinCon = new JLabel();
```

Webgrafía

- Para ver API’s de java:
  Java API
- El copiloto de confianza:
  ChatGPT