IES Ribera del Tajo

PRÁCTICA 3.1
UNIDAD 3

DESARROLLO DE INTERFACES

Alejandro Sánchez Gil
2ºDAM CURSO 2024/2025

Castilla-La Mancha

ÍNDICE

Métodos clave

Podemos definir un método como un bloque de código que ejecuta instrucciones cuando se realiza una llamada a dicho método, entonces podemos definir los métodos clave como aquellos que se encargan de definir y establecer las características de nuestro programa.

Los métodos clave que nos permiten analizar y extraer el contenido de propiedades o atributos son aquellos que llamamos getters, que usualmente tienen una sintaxis parecida a "objeto.getCaracterística" cambiando "objeto" por el objeto del que queramos obtener un atributo, que en este caso sería donde sustituiríamos "característica" como por ejemplo: "cliente.getName()", donde obtendríamos un supuesto nombre de un cliente.

Por otro lado, los métodos clave que nos permiten establecer o modificar valores son los llamados "setters", son muy útiles en el código dinámico para por ejemplo modificar valores en registros con la entrada del usuario, usualmente su sintaxis es "objeto.setCaracterística", de la misma manera que en el anterior "objeto" hace referencia a una entidad que hemos creado previamente y "característica" al atributo que queremos modificar de dicho objeto, utilizando el mismo ejemplo de antes podríamos modificar el nombre de un supuesto cliente de la siguiente manera: "cliente.setName()".

Es muy importante que a la hora de programar tengamos en cuenta el ámbito en el que se encuentran estos métodos, ya que si los queremos referenciar desde otra clase de nuestro proyecto y están en privado con el ámbito "private" las otras clases del proyecto no podrán ver estos métodos, por lo que los métodos clave siempre deben estar en public.

Creación de logotipo en GIMP

A continuación vamos a crear un logotipo en la aplicación “GIMP[1]” haciendo uso del tutorial facilitado por el profesor.

Para ello abriremos GIMP y crearemos un archivo nuevo donde iniciaremos la creación de nuestro logo.

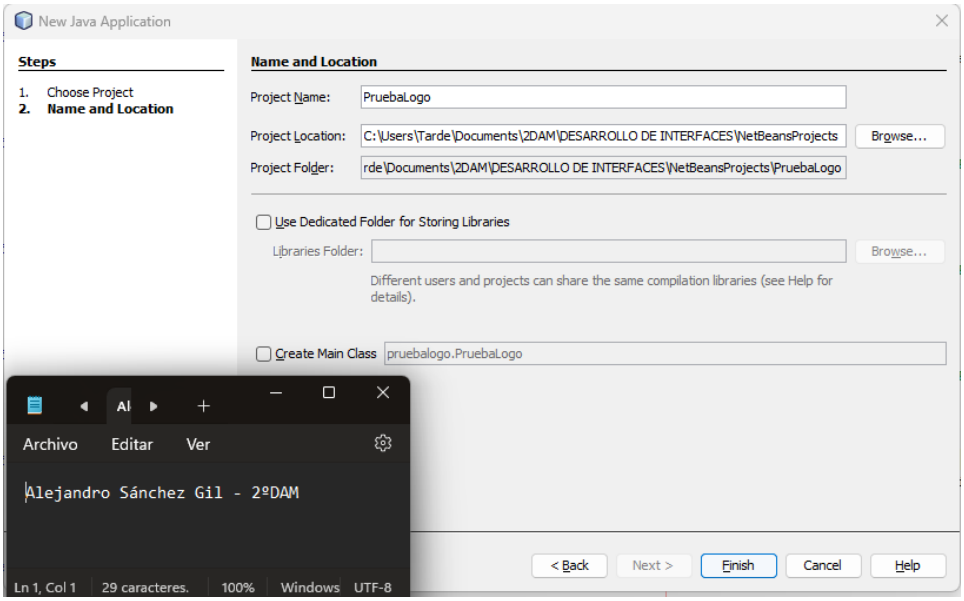
Aquí tenemos el logo que he realizado siguiendo el tutorial.

Implementación de logotipo en Netbeans

A continuación vamos a implementar nuestro logotipo en una aplicación de Netbeans, primero vamos a desarrollar un formulario, en este caso voy a desarrollar uno para registrarse en un torneo online de videojuegos.

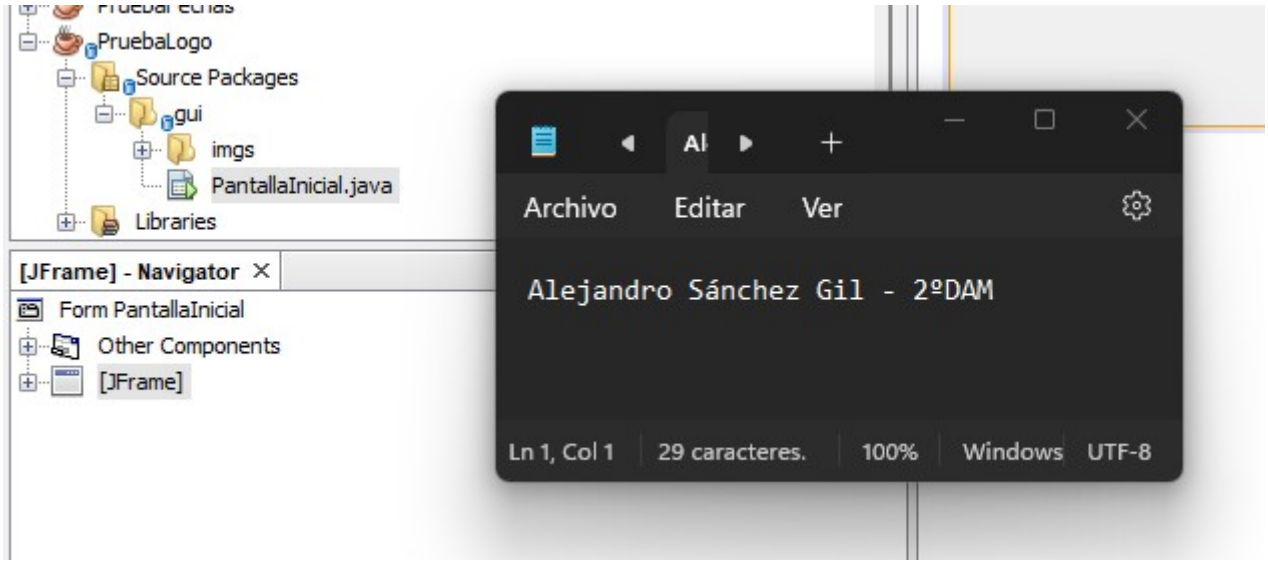
Primero para implementar nuestro logo en cualquier aplicación deberemos hacer lo siguiente:

1. Creamos el proyecto en el que vayamos a introducir nuestra imagen o logo:

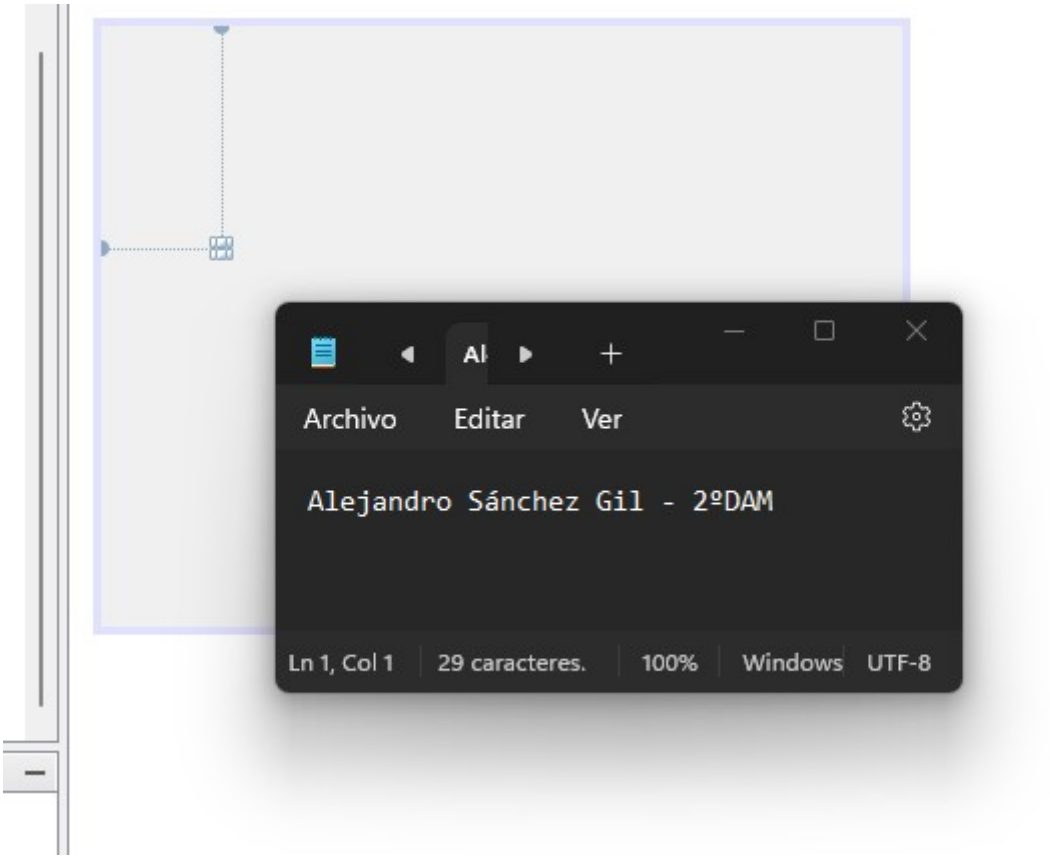


1 Gimp: Es una herramienta diseñada para la edición y creación de imágenes digitales.

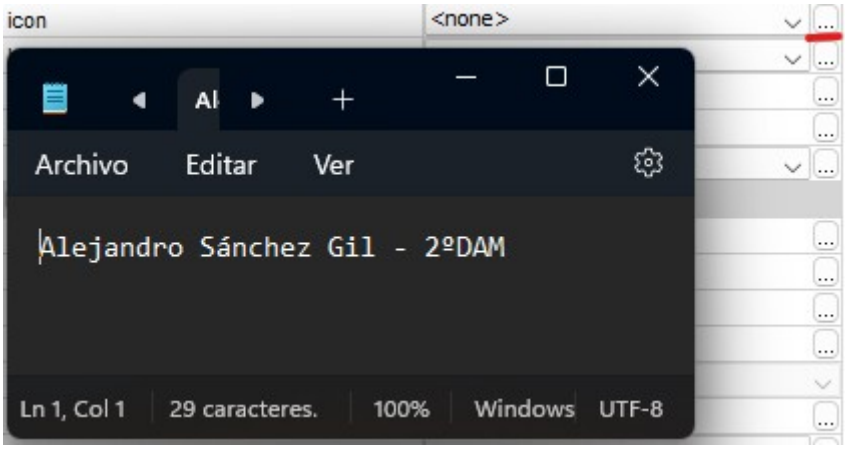
2. Creamos la organización de nuestro proyecto, ya que los elementos como las imágenes son elementos gráficos, sería correcto introducirlas en un paquete propio dentro del paquete “gui” o “interfaz”:



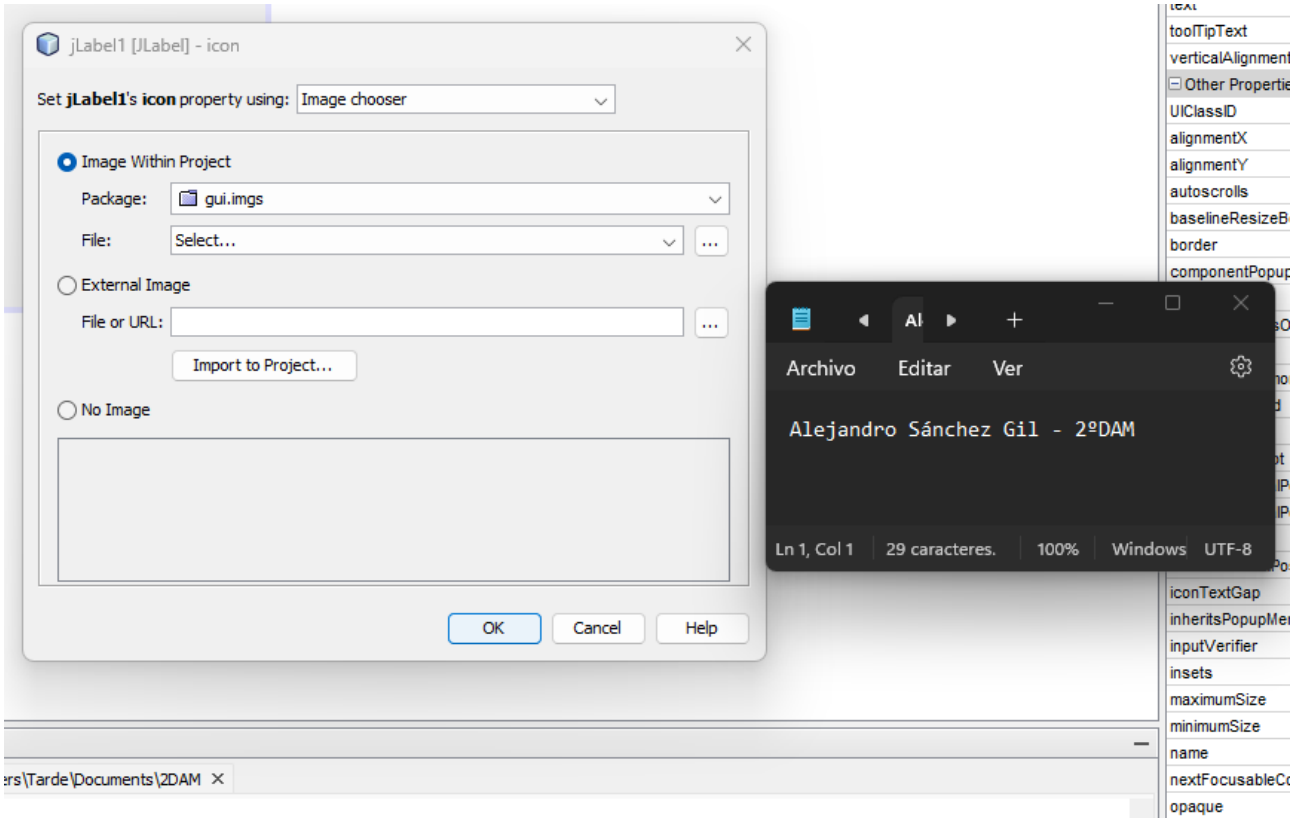
3. Como siempre crearemos un JFrame o JDialog y le introduciremos un jLabel al que le eliminaremos el texto:



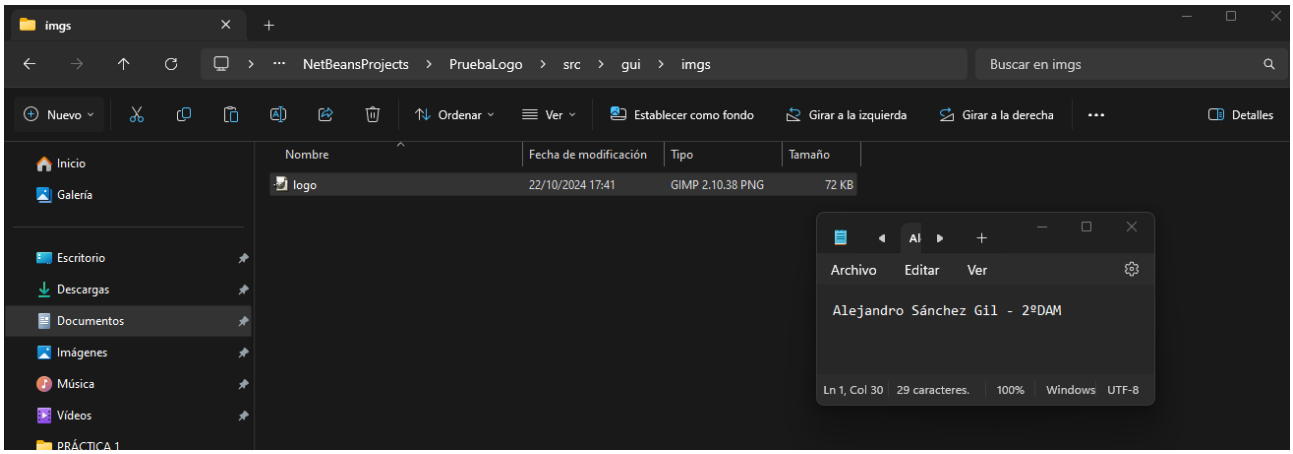
4. Ahora en las propiedades del jLabel encontraremos una de nombre “icon” hacemos clic en los tres puntos suspensivos:



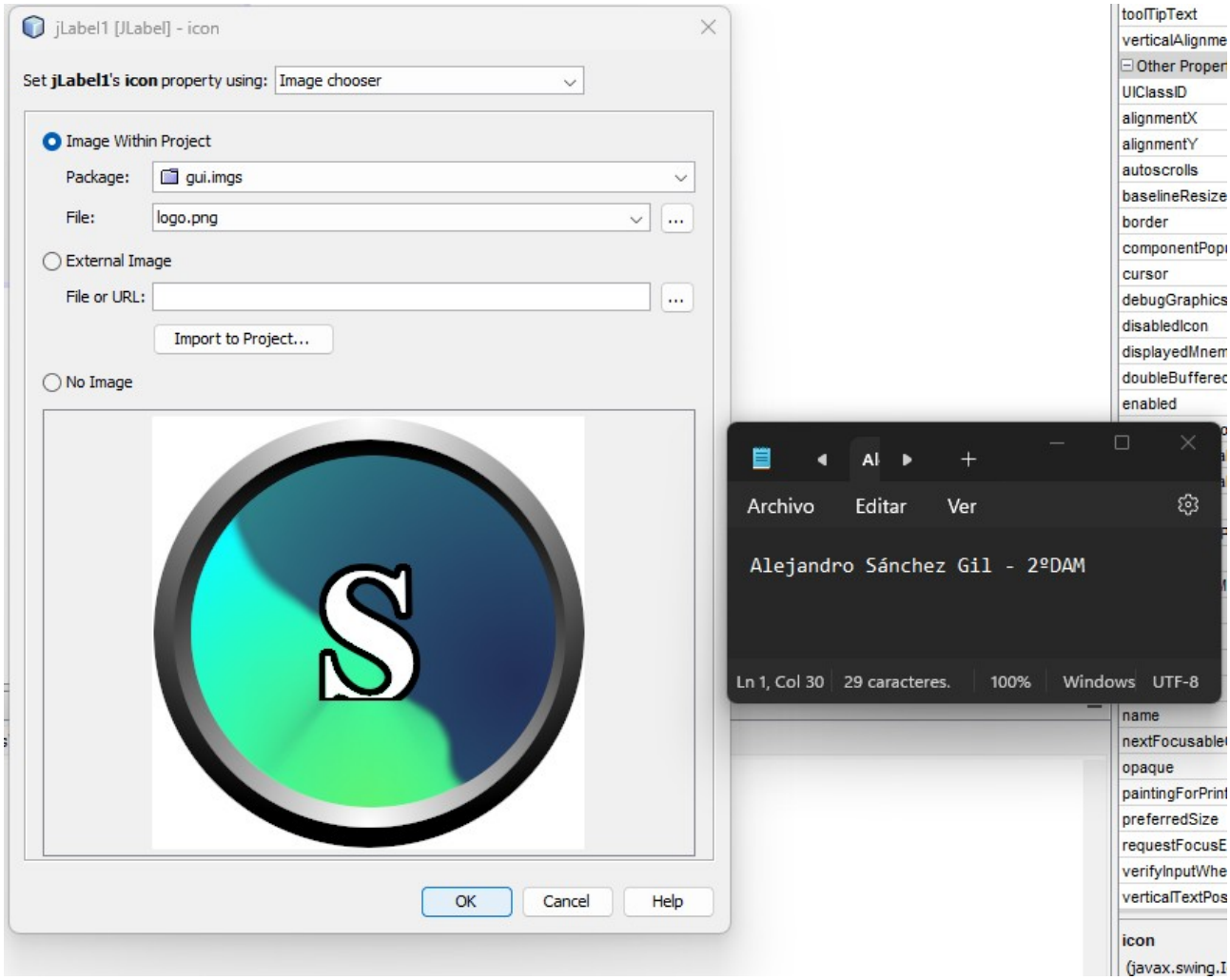
5. Ahora en la pestaña que se nos abrirá podremos seleccionar la imagen para ponerla, pero primero tenemos que poner la imagen en la ruta, así que hacemos eso:



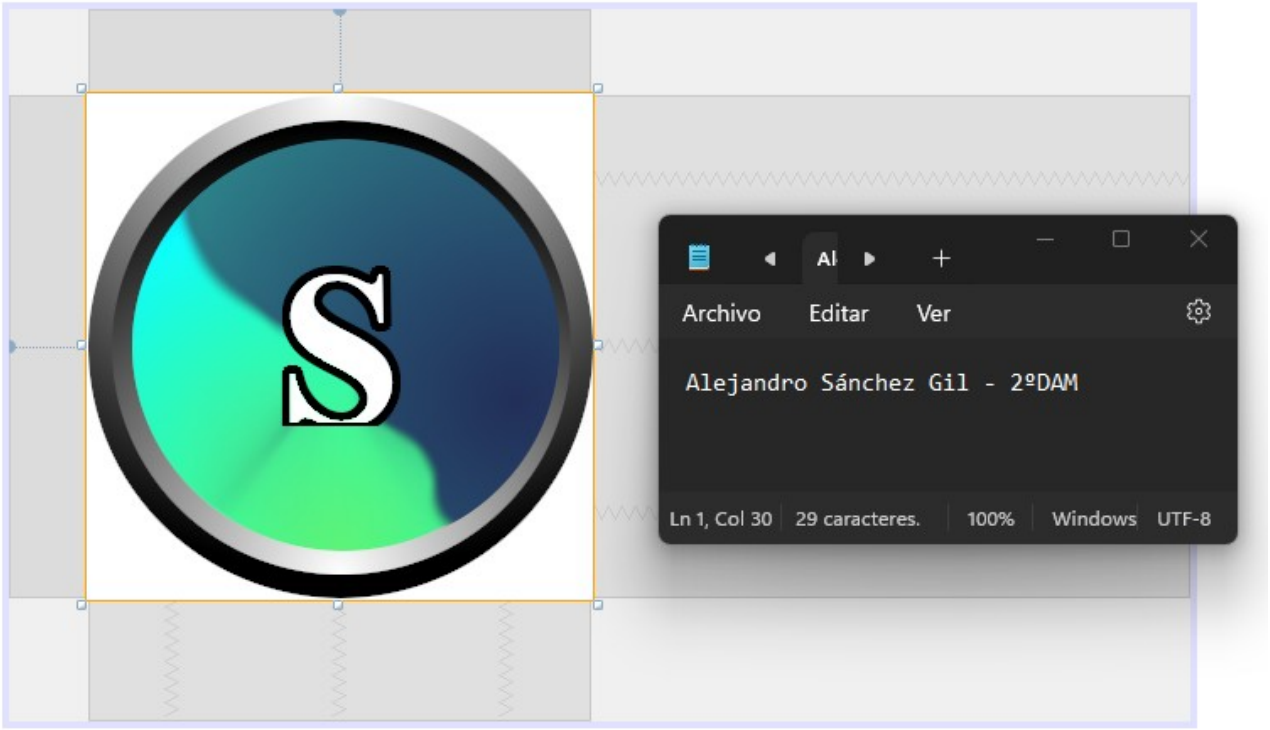
Asegúrate siempre de que la imagen que estás poniendo esté en el formato correcto, en este caso es png, ya que no queremos que tenga fondo:



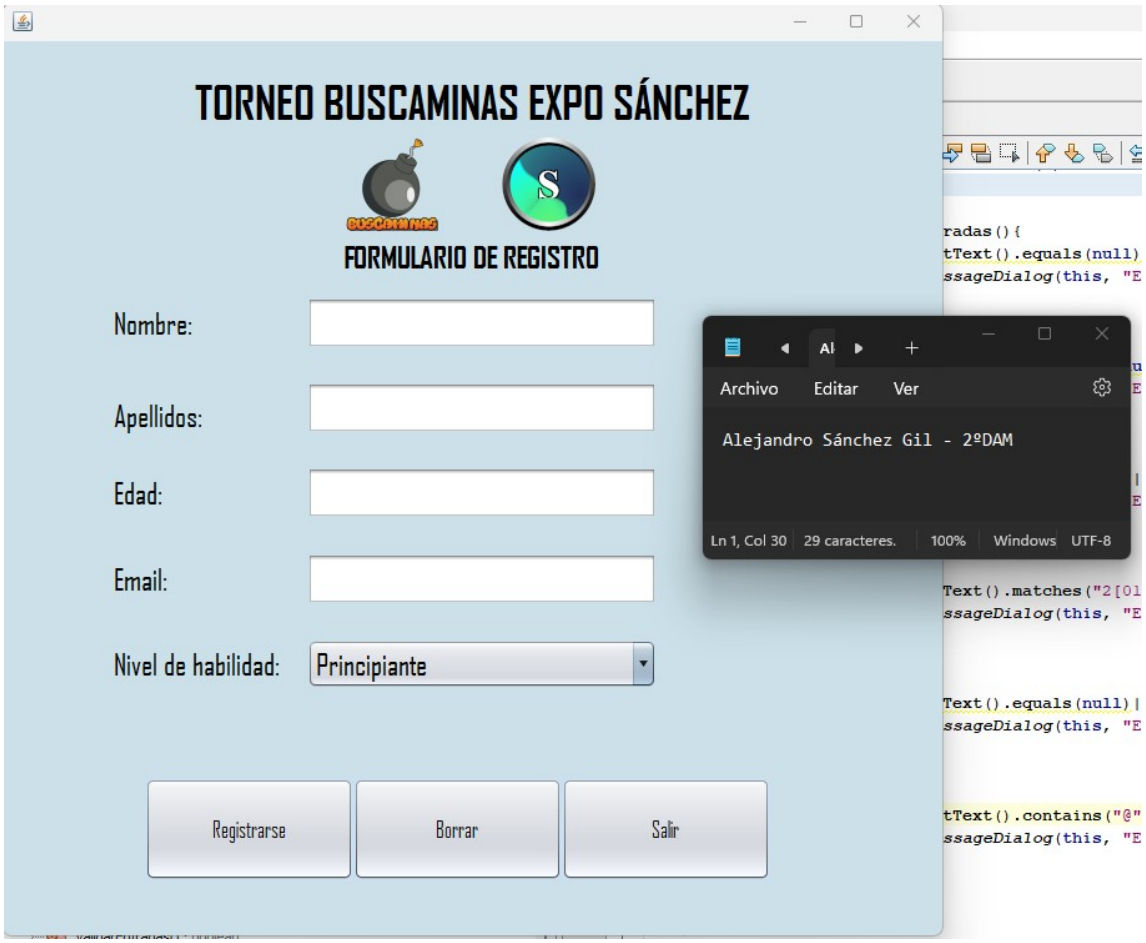
Ahora volvamos a NetBeans, como podremos ver, ahora si nos deja seleccionar nuestro logo y lo podremos ver previsualizado, ya solo nos queda hacer clic en OK:



Finalmente nuestro logo se habrá introducido en lugar del jLabel:



Ahora veamos la aplicación que yo he creado, como he dicho antes se trata de un formulario para un torneo online de buscaminas, después de ver la pantalla inicial veremos el código más a fondo:

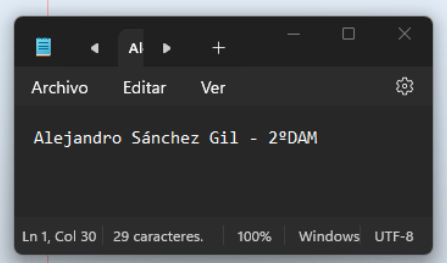


Como podemos ver en la imagen anterior, están incluídos los logos tanto del buscaminas como el logo que he creado anteriormente.

Aplicación en profundidad

Primero veamos los componentes que utiliza el formulario:

```
    // Variables declaration - do not modify
    private javax.swing.JButton jButtonBorrar;
    private javax.swing.JButton jButtonRegistrar;
    private javax.swing.JButton jButtonSalir;
    private javax.swing.JComboBox<String> jComboBoxNivel;
    private javax.swing.JLabel jLabelApellidos;
    private javax.swing.JLabel jLabelBuscaminasLogo;
    private javax.swing.JLabel jLabelEdad;
    private javax.swing.JLabel jLabelEmail;
    private javax.swing.JLabel jLabelLogoS;
    private javax.swing.JLabel jLabelNivel;
    private javax.swing.JLabel jLabelNombre;
    private javax.swing.JLabel jLabelTitulo;
    private javax.swing.JLabel jLabelTorneo;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField jTextFieldApellidos;
    private javax.swing.JTextField jTextFieldEdad;
    private javax.swing.JTextField jTextFieldEmail;
    private javax.swing.JTextField jTextFieldNombre;
    // End of variables declaration
}
```

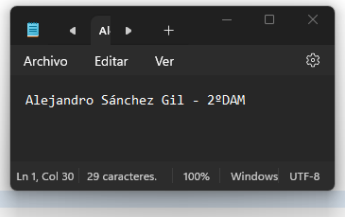


- Tres variables de tipo jButton de nombre “Borrar”, “Registrar” y “Salir” cada una con funciones a cumplir una vez se haya cumplido sus respectivos actionListeners[2], en este caso es el por defecto (“Click”).
- Una variable de tipo jComboBox de nombre “Nivel” que hace referencia al desplegable de la aplicación para seleccionar el nivel del usuario.
- Nueve variables de tipo jLabel, para indicar de manera visual ya sea información general como el titulo o información sobre los campos a rellenar o seleccionar.
- Cuatro campos de tipo jTextField donde el usuario introducirá sus datos personales para poder proceder con el registro del torneo.

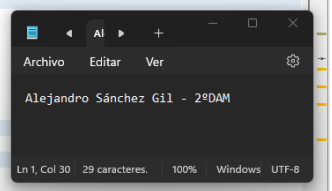
2 Los ActionListeners son métodos asociados a objetos que podemos decir que están “escuchando” a la espera de que se produzca un evento en la aplicación, para poder ejecutar las funciones a las que está asociado su objeto.

A continuación veremos en detalle las funciones del programa:

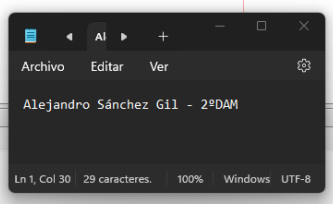
```java
package gui;

import javax.swing.JOptionPane;

/**
 *
 * @author Alejandro Sánchez Gil
 */
public class PantallaInicial extends javax.swing.JFrame {

    /**
     * Creates new form PantallaInicial
     */
    //Clase PantallaInicial con los métodos "initComponents()" que inicia todos los componentes de la aplicación y "getContentPane().setBackground" para cambiar el color
    //del fondo de la aplicación.
    public PantallaInicial() {
        initComponents();
        getContentPane().setBackground(new java.awt.Color(203, 224, 233));
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```




```java
    private void jTextFieldNombreActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    //Método ActionPerformed que tras haberse activado el ActionListener del botón "Registrar" se ejecutarán las siguientes ordenes.
    private void jButtonRegistrarActionPerformed(java.awt.event.ActionEvent evt) {
        //Comprobamos si las entradas son válidas con el método "validarEntradas()".
        if(validarEntradas()){
            //En caso de que sea true, nos lanza un mensaje que nos indica un registro satisfactorio.
            JOptionPane.showMessageDialog(this, "Se ha registrado correctamente a "+jTextFieldNombre.getText()+
                " "+jTextFieldApellidos.getText()+" en el torneo en la dificultad: "+
                jComboBoxNivel.getSelectedItem(),"Registro completado",JOptionPane.INFORMATION_MESSAGE);
        }
    }
    //Método ActionPerformed que tras haberse activado el ActionListener del botón "Salir" se ejecutarán las siguientes ordenes.
    private void jButtonSalirActionPerformed(java.awt.event.ActionEvent evt) {
        //Salida del sistema, es decir cierra la aplicación.
        System.exit(0);
    }
    //Método ActionPerformed que tras haberse activado el ActionListener del botón "Limpiar" se ejecutarán las siguientes ordenes.
    private void jButtonBorrarActionPerformed(java.awt.event.ActionEvent evt) {
        //Establece todos los campos en los que se puede escribir a vacio y devuelve el nivel a "Principiante" que es el primer valor.
        jTextFieldApellidos.setText("");
        jTextFieldNombre.setText("");
        jTextFieldEdad.setText("");
        jTextFieldEmail.setText("");
        jComboBoxNivel.setSelectedIndex(0);
    }
```




```java
private boolean validarEntradas(){
    //Comprueba que el nombre no esté vacío y en caso contrario lanza un mensaje de error y devuelve falso como valor.
    if(jTextFieldNombre.getText().equals(null)||jTextFieldNombre.getText().equals("")){
        JOptionPane.showMessageDialog(this, "Error, el nombre no puede estar vacío", "Error.", JOptionPane.ERROR_MESSAGE);
        return false;
    }
    //Comprueba que el apellido no esté vacío y en caso contrario lanza un mensaje de error y devuelve falso como valor.
    if(jTextFieldApellidos.getText().equals(null)||jTextFieldApellidos.getText().equals("")){
        JOptionPane.showMessageDialog(this, "Error, el apellido no puede estar vacío", "Error.", JOptionPane.ERROR_MESSAGE);
        return false;
    }
    //Comprueba que la edad no esté vacío y en caso contrario lanza un mensaje de error y devuelve falso como valor.
    if(jTextFieldEdad.getText().equals(null)||jTextFieldEdad.getText().equals("")){
        JOptionPane.showMessageDialog(this, "Error, la edad no puede estar vacía", "Error.", JOptionPane.ERROR_MESSAGE);
        return false;
    }
    //Comprueba que la edad sea un número y en caso contrario lanza un mensaje de error y devuelve falso como valor.
    if(!jTextFieldEdad.getText().matches("2[01]?\\d|[1-9]?\\d")){
        JOptionPane.showMessageDialog(this, "Error, la edad debe ser un número", "Error.", JOptionPane.ERROR_MESSAGE);
        return false;
    }
    //Comprueba que el email no esté vacío y en caso contrario lanza un mensaje de error y devuelve falso como valor.
    if(jTextFieldEmail.getText().equals(null)||jTextFieldEmail.getText().equals("")){
        JOptionPane.showMessageDialog(this, "Error, el email no puede estar vacío", "Error.", JOptionPane.ERROR_MESSAGE);
        return false;
    }
    //Comprueba que el email contenga "@" y ".", en caso contrario lanza un mensaje de error y devuelve falso como valor.
    if(!jTextFieldEmail.getText().contains("@")||!jTextFieldEmail.getText().contains(".")){
        JOptionPane.showMessageDialog(this, "Error, introduce un dominio de email correcto", "Error.", JOptionPane.ERROR_MESSAGE);
        return false;
    }

    return true;
}
```



Figuras geométricas

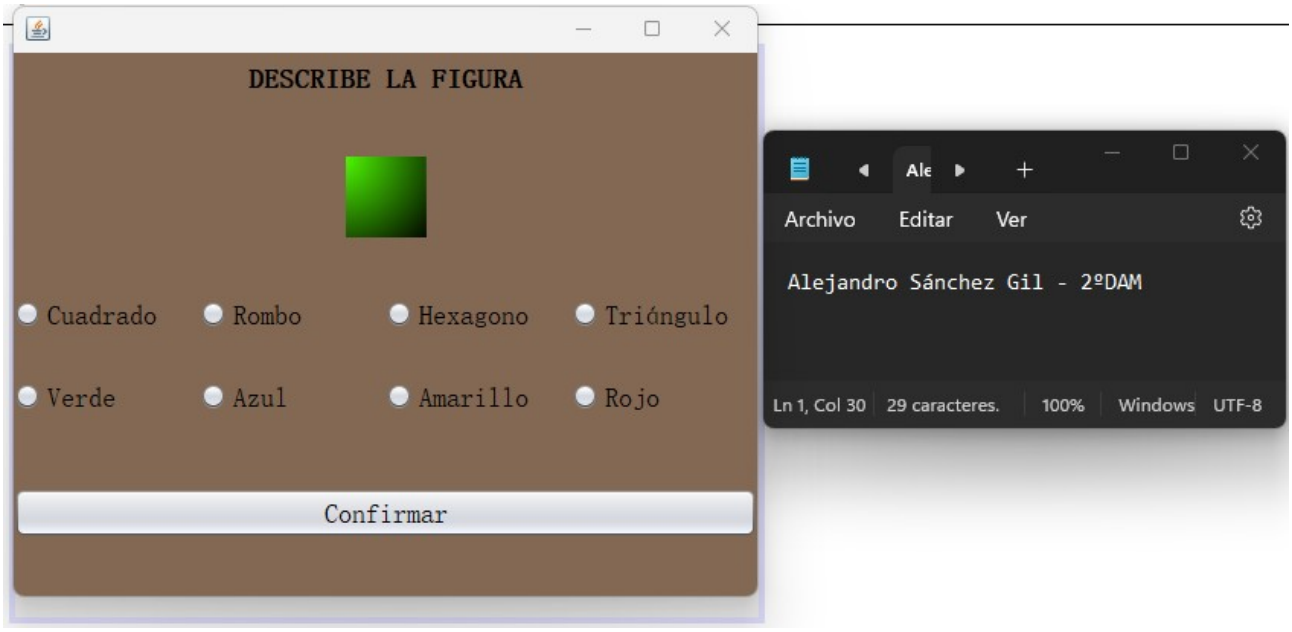
A continuación se nos solicita que creemos cuatro figuras geométricas distintas en GIMP también siguiendo un tutorial facilitado por el profesor, para posteriormente introducirlas en una interfaz que nosotros mismos diseñemos, veamos entonces las figuras que he creado:

Como podemos ver son un cuadrado verde, un hexágono rojo, un triángulo azul y un rombo amarillo.

Implementación en Netbeans

A continuación tendremos que implementar estas figuras en un proyecto de netbeans para ello yo he decidido hacer un juego donde el usuario debe de seleccionar correctamente cada figura y su color.

Veamos la aplicación:

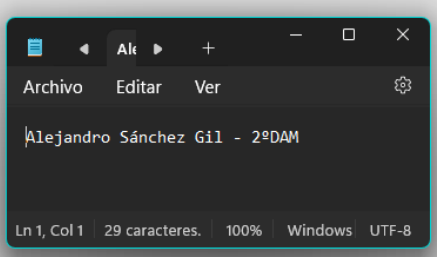


Como podemos ver se trata de una aplicación simple con distintos elementos.

Ahora veamos la aplicación en profundidad:

Aquí tenemos las variables del proyecto:

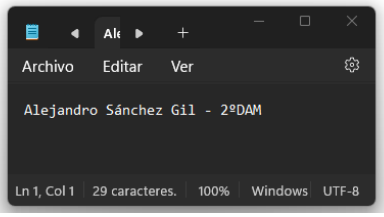
```
// Variables declaration - do not modify
private javax.swing.ButtonGroup buttonGroupColor;
private javax.swing.ButtonGroup buttonGroupFigura;
private javax.swing.JButton jButtonConfirmar;
private javax.swing.JLabel jLabelCuadrado;
private javax.swing.JLabel jLabelHexagono;
private javax.swing.JLabel jLabelRombo;
private javax.swing.JLabel jLabelSalida;
private javax.swing.JLabel jLabelTitulo;
private javax.swing.JLabel jLabelTriangulo;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JRadioButton jRadioButtonAmarillo;
private javax.swing.JRadioButton jRadioButtonAzul;
private javax.swing.JRadioButton jRadioButtonCuadrado;
private javax.swing.JRadioButton jRadioButtonHexagono;
private javax.swing.JRadioButton jRadioButtonRojo;
private javax.swing.JRadioButton jRadioButtonRombo;
private javax.swing.JRadioButton jRadioButtonTriangulo;
private javax.swing.JRadioButton jRadioButtonVerde;
// End of variables declaration
```



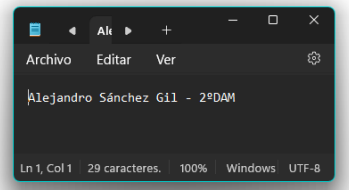
- Dos buttonGroups que agrupan o las figuras o los colores.
- Un botón confirmar para confirmar la respuesta y que se cumplan las funciones establecidas en el botón.
- Seis jLabels, de los cuales cuatro son sustituidos por las figuras geométricas, mientras que las otras dos son el título y la salida para indicar al usuario si es correcto o no.
- Tres jPanels para organizar el proyecto.
- Ocho radioButtons con las diferentes opciones para combinar y mostrar una respuesta.

Ahora veamos los métodos de la aplicación:

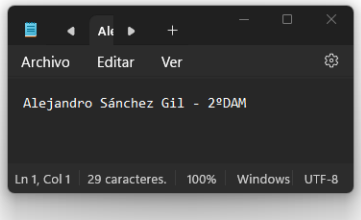
```
* @author Alejandro Sánchez Gil
*/
public class PantallaInicial extends javax.swing.JFrame {
    //Inicializamos variable contador que utilizaremos para igualar los procesos.
    int contador=0;
    /** Creates new form PantallaInicial */
    //Clase y atributos de la clase que iuncluyen el cambio de los colores del fondo de la app, y los paneles, al igual que esconder el resto de figuras.
    public PantallaInicial() {
        initComponents();
        getContentPane().setBackground(new java.awt.Color(131, 105, 83));
        jPanel1.setBackground(new java.awt.Color(131, 105, 83));
        jPanel2.setBackground(new java.awt.Color(131, 105, 83));
        jPanel3.setBackground(new java.awt.Color(131, 105, 83));
        jLabelHexagono.setVisible(false);
        jLabelRombo.setVisible(false);
        jLabelTriangulo.setVisible(false);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
```



```java
//Función que se ejecuta al pulsar el botón "Confirmar", se trata de un comprobador de la variable contador, que comprueba si es = a 0, 1, 2 o 3, en caso de que si
//esconde la figura actual y muestra la siguiente siempre que el usuario haya introducido la respuesta correcta, en caso contrario no avanza.
    private void jButtonConfirmarActionPerformed(java.awt.event.ActionEvent evt) {
        if(contador==0){
            if(jRadioButtonCuadrado.isSelected() && jRadioButtonVerde.isSelected()){
                jLabelCuadrado.setVisible(false);
                jLabelTriangulo.setVisible(true);
                jLabelSalida.setText("¡Correcto!");
                jLabelSalida.setForeground(Color.green);
                contador++;
            }else{
                jLabelSalida.setText("¡Incorrecto, fijate bien!");
                jLabelSalida.setForeground(Color.red);
            }

        }
        else if(contador==1){
            if(jRadioButtonTriangulo.isSelected() && jRadioButtonAzul.isSelected()){
                jLabelTriangulo.setVisible(false);
                jLabelRombo.setVisible(true);
                jLabelSalida.setText("¡Correcto!");
                jLabelSalida.setForeground(Color.green);
                contador++;
            }else{
                jLabelSalida.setText("¡Incorrecto, fijate bien!");
                jLabelSalida.setForeground(Color.red);
            }
        }
        else if(contador==2){
            if(jRadioButtonRombo.isSelected() && jRadioButtonAmarillo.isSelected()){
                jLabelRombo.setVisible(false);
                jLabelHexagono.setVisible(true);
                jLabelSalida.setText("¡Correcto!");
                jLabelSalida.setForeground(Color.green);
                contador++;
```




```java
                contador++;
            }else{
                jLabelSalida.setText("¡Incorrecto, fijate bien!");
                jLabelSalida.setForeground(Color.red);
            }
        }
        else if(contador==3){
            if(jRadioButtonHexagono.isSelected() && jRadioButtonRojo.isSelected()){
                jPanel1.setVisible(false);
                jLabelSalida.setText("¡Correcto!");
                jLabelSalida.setForeground(Color.green);
                contador++;
                jRadioButtonCuadrado.setEnabled(false);
                jRadioButtonRombo.setEnabled(false);
                jRadioButtonHexagono.setEnabled(false);
                jRadioButtonTriangulo.setEnabled(false);
                jRadioButtonVerde.setEnabled(false);
                jRadioButtonAzul.setEnabled(false);
                jRadioButtonAmarillo.setEnabled(false);
                jRadioButtonRojo.setEnabled(false);
                jButtonConfirmar.setEnabled(false);
            }else{
                jLabelSalida.setText("¡Incorrecto, fijate bien!");
                jLabelSalida.setForeground(Color.red);
            }
        }


    }
```



Y con esto habríamos completado la actividad 3.

MouseMotionListener y FocusListener

MouseMotionListener y FocusListener son interfaces que le permiten a los componentes de un proyecto responder a eventos realizados por el usuario, de la misma manera que EventActionListener “escucha” eventos realizados en el objeto en el que lo establezcamos, MouseMotionListener se utiliza para detectar y responder a los movimientos del ratón, mientras que FocusListener se utiliza para detectar sobre que elemento tenemos el foco, es decir cuando hacemos clic en dicho elemento por ejemplo, el foco pasa a ese objeto por lo general.

Métodos MouseMotionListener

Tiene dos métodos que son:

- mouseDragged(MouseEvent e): Al que se le llama cuando el usuario arrastra el ratón mientras mantiene un botón pulsado.
- mouseMoved(MouseEvent e): Al que se llama cuando el usuario mueve el ratón sin más.

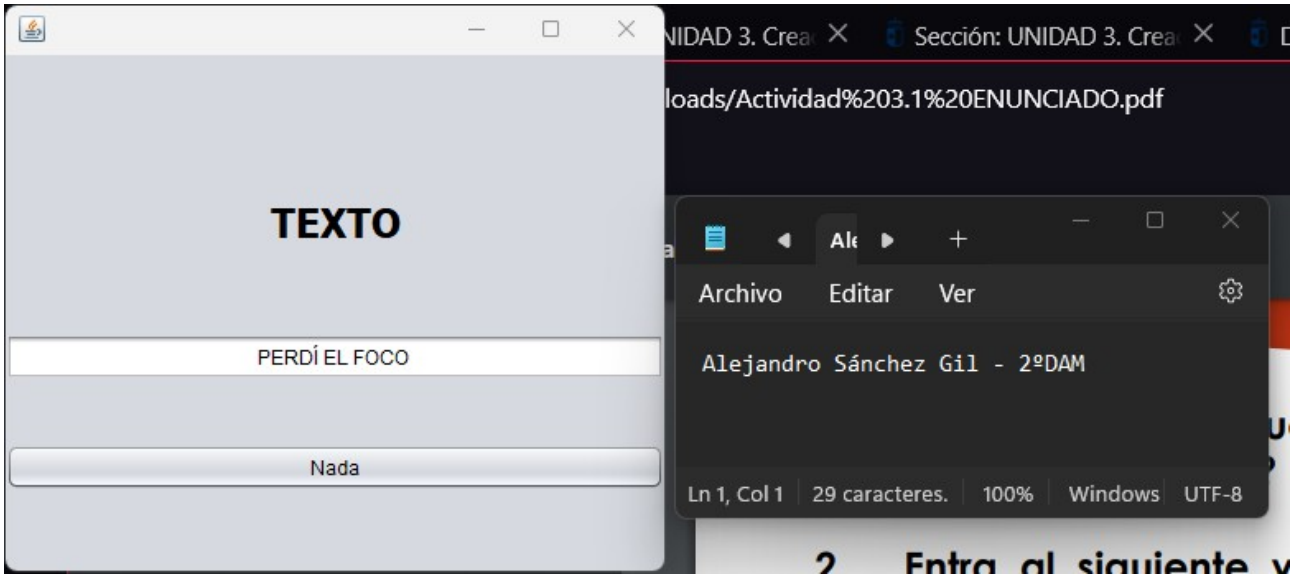
Métodos FocusListener

También tiene dos métodos que son:

- focusGained(FocusEvent e): Que se llama cuando el objeto obtiene el foco.
- focusLost(FocusEvent e): Que se llama cuando el objeto pierde el foco.

Implementación en NetBeans

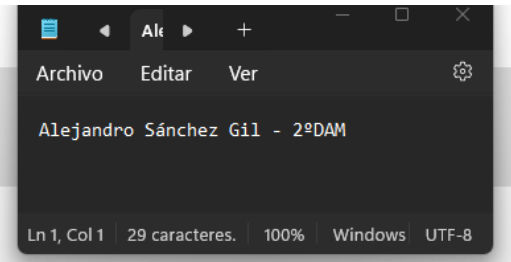
Para demostrar el funcionamiento de dichos elementos he creado una interfaz simple con tres elementos, un jLabel con un texto que cambia de color al pasar el ratón por encima y que al arrastrasrlo vuelve al negro, un jTextField que si tiene el foco indica que tiene el foco y en caso de perderlo indica que perdió el foco y un jButton que no tiene ninguna función asociada, que simplemente sirve para perder el foco del jTextField.



Ahora veamos las funciones dentro de la app:

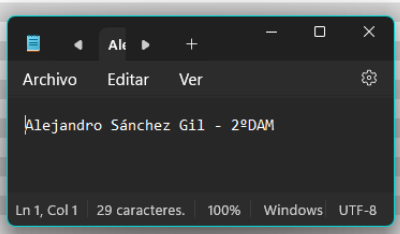
Variables:

```
    });
}

// Variables declaration - do not modify
private javax.swing.JButton jButtonNada;
private javax.swing.JLabel jLabelTexto;
private javax.swing.JTextField jTextFieldEjemplo;
// End of variables declaration
```



Funciones:

```
private void jLabelTextoMouseMoved(java.awt.event.MouseEvent evt) {
    jLabelTexto.setForeground(Color.yellow);
}

private void jLabelTextoMouseDragged(java.awt.event.MouseEvent evt) {
    jLabelTexto.setForeground(Color.black);
}

private void jTextFieldEjemploFocusGained(java.awt.event.FocusEvent evt) {
    jTextFieldEjemplo.setText("TENGO EL FOCO");
}

private void jTextFieldEjemploFocusLost(java.awt.event.FocusEvent evt) {
    jTextFieldEjemplo.setText("PERDÍ EL FOCO");
}

/**
```



Implementación a nivel empresarial

Considero que estas funciones pueden ser muy útiles en un proyecto importante, por ejemplo lo podríamos utilizar para un editor de diagramas donde el usuario al arrastrar un elemento pueda modificar su forma o cambiar su posición con MouseDragged, y podemos utilizar MouseMotionListener para medir con precisión la posición del ratón en tiempo real.

Y por ejemplo los FocusListeners los podemos utilizar para resaltar campos de formularios en una aplicación de registro.