



ACTIVIDAD 3.1

ALFONSO RINCÓN CUERVA
2º DAM
DESARROLLO DE INTERFACES

ÍNDICE

| | |
|--------------------|----|
| ACTIVIDAD 1 | 1 |
| ACTIVIDAD 2 | 2 |
| ACTIVIDAD 3 | 5 |
| ACTIVIDAD 4 | 9 |
| BIBLIOGRAFÍA | 13 |

ACTIVIDAD 1

Reflexiona sobre qué métodos claves permiten analizar el contenido de una propiedad o atributo. ¿Cuáles modifican su valor?

ANALIZAR:

Estos métodos vendrían a ser los Getters, que son los que permiten ver el valor de una propiedad o atributo. No modifican el estado del objeto, simplemente se encargan de obtener su valor y proporcionar información sobre su estado. Los componentes también lo tendrán, aunque dependiendo del componente en cuestión, utilizarán unos métodos u otros.

Por ejemplo:

- **getText():** para obtener el texto de algunos componentes.
- **isSelected():** para componentes como los CheckBox y los RadioButtons. Esto nos permitirá saber si el botón es seleccionado o no.
- **isEnabled():** indica si un componente está habilitado.

MODIFICAR:

Estos métodos son los que modifican el valor de una propiedad o componente. Son los Setters, y controlan cuándo y cómo se modifica, tanto un atributo como un componente.

Por ejemplo:

- **setText():** cambia el texto de un componente.
- **setEnabled():** habilita o deshabilita un componente.
- **setSelected():** sirve para definir si está o no seleccionado el componente en cuestión.

ACTIVIDAD 2

Entra al siguiente video de YouTube:

https://www.youtube.com/watch?v=0hH4iyjIMwc&ab_channel=ChepeGeek

Utilizando GIMP, crea un logotipo para tu interfaz parecido (no hace falta que sea exactamente ese), pero con la letra de tu primer apellido y el fondo del color que desees, excepto el mostrado en el video. Después, impleméntalo en una interfaz que tú diseñes en NetBeans 8.2 (por ejemplo, un formulario de inscripción a un gimnasio).

Se valorará la creatividad del diseño del logotipo y de la interfaz.

```

8 import java.awt.Color;
9 import javax.swing.BorderFactory;
10 import javax.swing.ImageIcon;
11 import javax.swing.JOptionPane;
12 import javax.swing.border.MatteBorder;
13
14 /**
15  *
16  * @author Alfonso
17  */
18 public class PantallaPrincipal extends javax.swing.JFrame {
19
20     /**
21      * Creates new form PantallaPrincipal
22      */
23     public PantallaPrincipal() {
24         initComponents();
25
26         // Borde para los JTextFields
27         MatteBorder bordeAbajo=BorderFactory.createMatteBorder(0, 0, 1, 0, Color.GRAY);
28         jTextFieldUsuario.setBorder(bordeAbajo);
29         jPasswordFieldContra.setBorder(bordeAbajo);
30
31         // Icono de la aplicación
32         this.setIconImage(new ImageIcon(getClass().getResource("/actividad3ejercicio2/images/logo.png")).getImage());
33     }
34
35     @SuppressWarnings("unchecked")
36     // Generated Code
37
38     // Hover para el botón
39     private void jLabel5MouseClicked(java.awt.event.MouseEvent evt) {
40         botonIniciarSesion.setBackground(new Color(4, 122, 79));
41     }
42
43     private void jLabel5MouseExited(java.awt.event.MouseEvent evt) {
44         botonIniciarSesion.setBackground(new Color(0, 168, 107));
45     }
46
47     // PLACEHOLDER PARA EL TEXT FIELD DE USUARIO
48     private void jTextFieldUsuarioFocusGained(java.awt.event.FocusEvent evt) {
49         String texto=jTextFieldUsuario.getText();
50         if(texto.equals("Insertar usuario...")) {
51             jTextFieldUsuario.setText("");
52             jTextFieldUsuario.setForeground(Color.BLACK);
53         }
54     }
55
56     private void jTextFieldUsuarioFocusLost(java.awt.event.FocusEvent evt) {
57         String texto=jTextFieldUsuario.getText();
58         if(texto.equals("")) {
59             jTextFieldUsuario.setText("Insertar usuario...");
60             jTextFieldUsuario.setForeground(new Color(153,153,153));
61         }
62     }
63
64     public boolean validarFormulario() {
65         String usu=jTextFieldUsuario.getText();
66         String contra=jPasswordFieldContra.getText();
67
68         if(!usu.equals("") && !contra.equals("") && !usu.equals("Insertar usuario...") && !contra.equals("jPasswordField1")) {
69             JOptionPane.showMessageDialog(this, "Sesión iniciada con éxito");
70             return true;
71         } else {
72             JOptionPane.showMessageDialog(this, "Datos incorrectos");
73             return false;
74         }
75     }
76
77     // CLICK Y VALIDAR INICIO DE SESIÓN
78     private void jLabel5MouseClicked(java.awt.event.MouseEvent evt) {
79         validarFormulario();
80     }
81
82     // PLACEHOLDER CONTRASEÑA
83     private void jPasswordFieldContraFocusGained(java.awt.event.FocusEvent evt) {
84         String texto=jPasswordFieldContra.getText();
85         if(texto.equals("jPasswordField1")) {
86             jPasswordFieldContra.setText("");
87             jPasswordFieldContra.setForeground(Color.BLACK);
88         }
89     }
90
91     private void jPasswordFieldContraFocusLost(java.awt.event.FocusEvent evt) {
92         String texto=jPasswordFieldContra.getText();
93         if(texto.equals("")) {
94             jPasswordFieldContra.setText("jPasswordField1");
95             jPasswordFieldContra.setForeground(new Color(153,153,153));
96         }
97     }
98
99 }

```

```

284 }
285
286 /**
287  * @param args the command line arguments
288  */
289 public static void main(String args[]) {
290     /* Set the Nimbus look and feel */
291     /* Look and feel setting code (optional)
292
293     /* Create and display the form */
294     java.awt.EventQueue.invokeLater(new Runnable() {
295         public void run() {
296             new PantallaPrincipal().setVisible(true);
297         }
298     });
299 }
300
301 // Variables declaration - do not modify
302 private javax.swing.JPanel BotonIniciarSesion;
303 private javax.swing.JPanel Contenedor;
304 private javax.swing.JPanel ContenedorTextFields;
305 private javax.swing.JLabel Logo;
306 private javax.swing.JLabel jLabel2;
307 private javax.swing.JLabel jLabel3;
308 private javax.swing.JLabel jLabel4;
309 private javax.swing.JLabel jLabel5;
310 private javax.swing.JPanel jPanelVerde;
311 private javax.swing.JPasswordField jPasswordFieldContra;
312 private javax.swing.JTextField jTextFieldUsuario;

```

FUNCIONAMIENTO

Esta interfaz consiste en un inicio de sesión para una aplicación de una empresa de Tecnología. Validará si el usuario y la contraseña son válidos (No está vacío el campo), y una vez lo sean, lanzará una alerta para confirmar que todo está correcto.



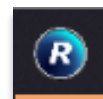
EXPLICACIÓN

Para empezar, se agregan los diferentes componentes al JFrame, entre los cuales tenemos:

- 1 JPanel que actúa como contenedor general de toda la interfaz.
- 1 JPanel para crear la parte verde.
- Varios JLabel para el texto, así como otro para la imagen de la interfaz.
- 1 JPanel junto con un JLabel, para crear el botón.
- 1 JTextField y 1 JPasswordField, para poder insertar los datos.

Una vez agregados los componentes, se les darán sus respectivos estilos. Al JFrame se le quitará que pondrá el redimensionable en false, y el locationByPlatform en true, para que no se pueda redimensionar, y al abrir la interfaz, se **posicione** mejor en la pantalla.

Después del initComponents(), se le aplicará un icono a la interfaz, mediante el **setIconImage**. Este icono fue creado con GIMP. También se les cambiará el estilo a los JTextField, creando un nuevo borde inferior con la clase **MatteBorder**.



Es el momento de empezar a programar su funcionalidad. Para el botón, agregamos 2 listeners, **MouseEntered** y **MouseExited**, en los cuales cambiamos el color de fondo de este. Con esto, conseguimos crear un efecto de Hover de forma manual. He optado por usar esto en lugar de JButtons, para lograr un diseño más acorde al resto de la interfaz.

Ahora crearemos un efecto de **Placeholder** en los `TextField`. Debido a que Java Swing no tiene esta funcionalidad, lo crearemos utilizando los `Listeners` de **FocusGained** y **FocusLost**. En el momento en el que el `TextField` tenga el foco, se vaciará su contenido, y cuando lo pierda, se rellenará con un contenido por defecto (Solo en caso de que esté vacío).

Se crea una función **validarFormulario**, la cual se llama posteriormente al pulsar el botón. Esta función, validará que el contenido de ambos `TextField` no esté vacío, así como tampoco sea el texto que tiene por defecto. Si cumple estas características, devolverá `true` y se mostrará un dialog que confirma que has iniciado sesión correctamente. En caso contrario, devolverá `false`, y el dialog que mostrará dirá que la información está mal.

ICONO CREADO CON GIMP:



ACTIVIDAD 3

Entra al siguiente vídeo de YouTube:

https://www.youtube.com/watch?v=gk6qR5JK6uA&ab_channel=Aulaenlanube

Utilizando GIMP crea 4 figuras geométricas distintas (por ejemplo, un cuadrado, un triángulo, un rectángulo y un rombo). Después, impleméntalo en una interfaz que tú diseñes en NEATBEANS 8.2 (por ejemplo, un examen test de matemáticas de identificar figuras geométricas). Se valorará la creatividad del diseño de las figuras y de la interfaz.

JFRAME

```

13  */
14  public class PantallaPrincipal extends javax.swing.JFrame {
15
16      /**
17       * Creates new form PantallaPrincipal
18       */
19      public PantallaPrincipal() {
20          initComponents();
21      }
22
23      /**
24       * This method is called from within the constructor to initialize the form.
25       * WARNING: Do NOT modify this code. The content of this method is always
26       * regenerated by the Form Editor.
27       */
28      @SuppressWarnings("unchecked")
29      // Generated Code
30
31      private void jLabel2MouseClicked(java.awt.event.MouseEvent evt) {
32          Examen dialog=new Examen(this, true);
33          dialog.setVisible(true);
34      }
35
36      // HOVER DEL BOTÓN
37      private void jLabel2MouseEntered(java.awt.event.MouseEvent evt) {
38          BotonEmpezar.setBackground(new Color(17, 98, 115));
39      }
40
41      private void jLabel2MouseExited(java.awt.event.MouseEvent evt) {
42          BotonEmpezar.setBackground(new Color(10,131,156));
43      }
44
45      /**
46       * @param args the command line arguments
47       */
48      public static void main(String args[]) {
49          /* Set the Nimbus look and feel */
50          // Look and feel setting code (optional)
51
52          /* Create and display the form */
53          java.awt.EventQueue.invokeLater(new Runnable() {
54              public void run() {
55                  new PantallaPrincipal().setVisible(true);
56              }
57          });
58      }
59
60      // Variables declaration - do not modify
61      private javax.swing.JPanel BotonEmpezar;
62      private javax.swing.JPanel Contenedor;
63      private javax.swing.JLabel jLabel1;
64      private javax.swing.JLabel jLabel2;
65      // End of variables declaration
66  }

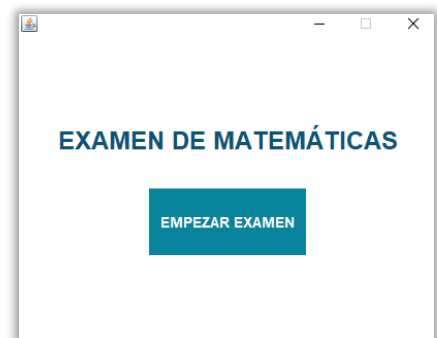
```

FUNCIONAMIENTO

Esta es la pantalla principal, es bastante básica. Tiene un botón, que, al pulsarlo, te llevará al examen.

EXPLICACIÓN

Está compuesto por un JLabel para el texto, y un JPanel que forma el botón. Tiene estilos de MouseEntered y MouseExited para simular el hover.



Una vez se pulsa el botón, se abrirá el Dialog que contiene el examen.

DIALOG

```

16 public class Examen extends javax.swing.JDialog {
17
18     /**
19      * Creates new form Examen
20      */
21     public Examen(java.awt.Frame parent, boolean modal) {
22         super(parent, modal);
23         initComponents();
24     }
25
26     /**
27      * This method is called from within the constructor to initialize the form.
28      * WARNING: Do NOT modify this code. The content of this method is always
29      * regenerated by the Form Editor.
30      */
31     @SuppressWarnings("unchecked")
32     // Generated Code
33
34     // MOVERNOS DE UNA PESTAÑA A OTRA
35     private void jLabel1MouseClicked(java.awt.event.MouseEvent evt) {
36         jTablebedPane.setSelectedIndex(1);
37     }
38
39     private void jLabel2MouseClicked(java.awt.event.MouseEvent evt) {
40         jTablebedPane.setSelectedIndex(2);
41     }
42
43     private void jLabel3MouseClicked(java.awt.event.MouseEvent evt) {
44         jTablebedPane.setSelectedIndex(3);
45     }
46
47     public void deshabilitarBotones() {
48         jButtonCirculo1.setEnabled(false);
49         jButtonCirculo2.setEnabled(false);
50         jButtonCirculo3.setEnabled(false);
51         jButtonCirculo4.setEnabled(false);
52         jButtonCuadrado1.setEnabled(false);
53         jButtonCuadrado2.setEnabled(false);
54         jButtonCuadrado3.setEnabled(false);
55         jButtonCuadrado4.setEnabled(false);
56         jButtonRombo1.setEnabled(false);
57         jButtonRombo2.setEnabled(false);
58         jButtonRombo3.setEnabled(false);
59         jButtonRombo4.setEnabled(false);
60         jButtonTriangulo1.setEnabled(false);
61         jButtonTriangulo2.setEnabled(false);
62         jButtonTriangulo3.setEnabled(false);
63         jButtonTriangulo4.setEnabled(false);
64     }
65
66     private void jLabel4MouseClicked(java.awt.event.MouseEvent evt) {
67         int cont=0;
68         int contCorrectas=0;
69
70         // Primera pregunta
71         if(jButtonCirculo1.isSelected() || jButtonCuadrado1.isSelected() || jButtonRombo1.isSelected() || jButtonTriangulo1.isSelected()) {
72             cont++;
73             if(jButtonCirculo1.isSelected()) {
74                 contCorrectas++;
75             }
76         }
77
78         // Segunda pregunta
79         if(jButtonCirculo2.isSelected() || jButtonCuadrado2.isSelected() || jButtonRombo2.isSelected() || jButtonTriangulo2.isSelected()) {
80             cont++;
81             if(jButtonCirculo2.isSelected()) {
82                 contCorrectas++;
83             }
84         }
85
86         // Tercera pregunta
87         if(jButtonCirculo3.isSelected() || jButtonCuadrado3.isSelected() || jButtonRombo3.isSelected() || jButtonTriangulo3.isSelected()) {
88             cont++;
89             if(jButtonCirculo3.isSelected()) {
90                 contCorrectas++;
91             }
92         }
93
94         // Cuarta pregunta
95         if(jButtonCirculo4.isSelected() || jButtonCuadrado4.isSelected() || jButtonRombo4.isSelected() || jButtonTriangulo4.isSelected()) {
96             cont++;
97             if(jButtonCirculo4.isSelected()) {
98                 contCorrectas++;
99             }
100         }
101     }
102 }

```



```

557     }
558
559     // Condicional para verificar que se respondan todas las preguntas
560     if(cont == 4) {
561         jTabbedPane.setSelectedIndex(4);
562
563         jLabelResultado.setText(contCorrectas+"/"+4);
564         if(contCorrectas > 2) {
565             jLabelResultado.setForeground(Color.GREEN);
566         } else {
567             jLabelResultado.setForeground(Color.RED);
568         }
569
570         deshabilitarBotones();
571     } else {
572         JOptionPane.showMessageDialog(this, "Hay preguntas sin responder");
573     }
574 }
575
576 // EVENTOS DEL HOVER DEL BOTÓN
577 private void jLabel1MouseClicked(java.awt.event.MouseEvent evt) {
578     IrAPregunta2.setBackground(new Color(196, 14, 14));
579 }
580
581 private void jLabel1MouseExited(java.awt.event.MouseEvent evt) {
582     IrAPregunta2.setBackground(new Color(153,0,0));
583 }
584
585 private void jLabel2MouseClicked(java.awt.event.MouseEvent evt) {
586     IrAPregunta3.setBackground(new Color(196, 14, 14));
587 }
588
589 private void jLabel2MouseExited(java.awt.event.MouseEvent evt) {
590     IrAPregunta3.setBackground(new Color(153,0,0));
591 }
592
593 // Variables declaration - do not modify
594 private javax.swing.JPanel ContainerPregunta1;
595 private javax.swing.JPanel ContainerPregunta2;
596 private javax.swing.JPanel ContainerPregunta3;
597 private javax.swing.JPanel ContainerPregunta4;
598 private javax.swing.JPanel IrAPregunta2;
599 private javax.swing.JPanel IrAPregunta3;
600 private javax.swing.JPanel IrAPregunta4;
601 private javax.swing.JPanel IrResultado;

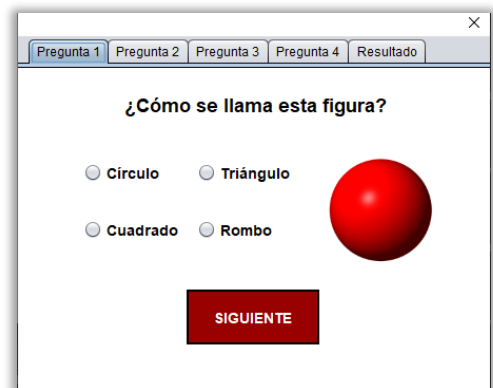
```

FUNCIONAMIENTO

Una vez empezado el examen, este `JDial` nos mostrará 4 preguntas, que debemos responder con `RadioButtons`. En cada pestaña, hay un botón de siguiente, que, al pulsarlo, te permite ir a la siguiente pregunta.

En la última pestaña, tenemos un botón de RESULTADO, el cual, al pulsarlo, en caso de que haya preguntas sin responder, lanzará un `Dialog`.

Una vez respondidas todas las preguntas, en la pestaña de Resultado aparecerán cuántas preguntas has acertado.



EXPLICACIÓN

Los componentes que forman este `JDial` son los siguientes:

- 1 `JTabbedPane`, que será de contendor para poner las diferentes pestañas.
- 1 `JPanel` por pestaña.
- Varios `JLabels` para las preguntas.
- Varios `JLabels` para cada uno de los dibujos. Estos tendrán un icono, que será el dibujo. Fueron dibujados con GIMP.
- `JPanels` que junto con `JLabel`, forman botones.
- 1 `JLabel` para mostrar el resultado.

- 4 RadioButtons por cada pestaña, además de un ButtonGroup para cada pestaña.

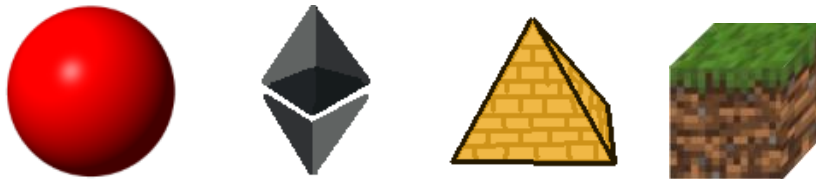
No será redimensionable, y también tendrá activado el **locationByPlatform**.

Cada botón tendrá unos Listener de **MouseEntered** y **MouseExited**, para crear un Hover. Además, tendrán uno de Click, para que cada uno al pulsar te lleve a la pestaña siguiente.

El último botón, al pulsarlo, verificará que se han respondido a todas las preguntas mediante un contador. Esto lo hará mediante varias condicionales, y en caso de que no estén todas respondidas, lanzará un Dialog. También tendrá la variable **contCorrectas**, que es la que nos dirá la cantidad de respuestas correctas que ha respondido el usuario. Este resultado aparecerá en la pestaña de Resultados.

Cuando hayas respondido todas las preguntas y hayas obtenido el resultado al pulsar en el botón de RESULTADO, se ejecutará el método **deshabilitarBotones**, que pondrá todos los botones en **setEnabled(false)**, para que no se pueda volver a responder.

FIGURAS CREADAS CIN GIMP:



ACTIVIDAD 4

Investiga e implementa en una interfaz en NEATBEANS 8.2 tanto el **MouseMotionListener** como el **FocusListener**. Si el proyecto te lo encargara una empresa, ¿en qué tipo de interfaces resultarían interesantes utilizar cada uno de los Listeners mencionados?

MOUSE MOTION LISTENERS:

Estos sirven para detectar el movimiento del ratón y arrastres dentro de una aplicación. Se podrá utilizar en interfaces que requieran las siguientes funciones:

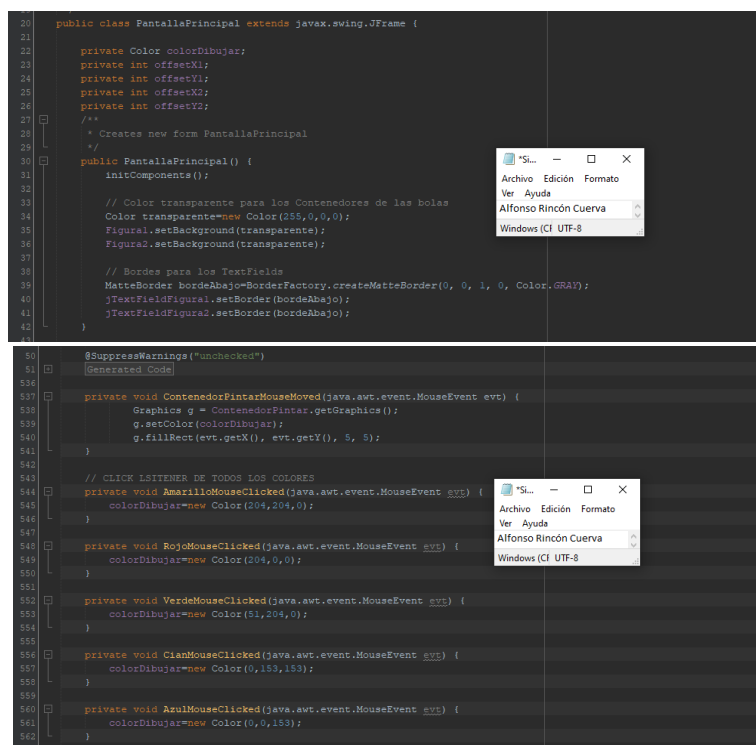
- Diseño gráfico y dibujo.
- Mapas interactivos, sobre los cuales puedes navegar arrastrando.
- Juegos en primera persona, basados en la orientación del ratón.
- Funciones de Arrastrar y Soltar elementos.
- Elementos que siguen a un cursor.

FOCUS LISTENER:

Algunos ejemplos de interfaces en las que podríamos utilizar este tipo de Listeners son los siguientes:

- Crear un placeholder, ya que Java Swing no tiene esta opción.
- Cambiar el estilo de un elemento en el momento en el que obtiene el foco.
- Aplicaciones que tiene múltiples ventanas-
- Sugerencias y autocompletado, en el momento en el que el elemento en cuestión recibe el foco.

INTERFAZ



```

20 public class PantallaPrincipal extends javax.swing.JFrame {
21
22     private Color colorDibujar;
23     private int offsetX;
24     private int offsetY;
25     private int offsetX2;
26     private int offsetY2;
27
28     /**
29      * Creates new form PantallaPrincipal
30      */
31     public PantallaPrincipal() {
32         initComponents();
33
34         // Color transparente para los Contenedores de las bolas
35         Color transparente=new Color(255,0,0,0);
36         Figura1.setBackground(transparente);
37         Figura2.setBackground(transparente);
38
39         // Bordes para los TextFields
40         MatteBorder bordeAbajo=BorderFactory.createMatteBorder(0, 0, 1, 0, Color.GRAY);
41         jTextFieldFigura1.setBorder(bordeAbajo);
42         jTextFieldFigura2.setBorder(bordeAbajo);
43     }
44
45     @SuppressWarnings("unchecked")
46     // Generated Code
47
48     private void ContenedorPintarMouseClicked(java.awt.event.MouseEvent evt) {
49         Graphics g = ContenedorPintar.getGraphics();
50         g.setColor(colorDibujar);
51         g.fillRect(evt.getX(), evt.getY(), 5, 5);
52     }
53
54     // CLICK LISTENER DE TODOS LOS COLORES
55     private void AmarilloMouseClicked(java.awt.event.MouseEvent evt) {
56         colorDibujar=new Color(255,255,0);
57     }
58
59     private void RojoMouseClicked(java.awt.event.MouseEvent evt) {
60         colorDibujar=new Color(255,0,0);
61     }
62
63     private void VerdeMouseClicked(java.awt.event.MouseEvent evt) {
64         colorDibujar=new Color(0,255,0);
65     }
66
67     private void CianMouseClicked(java.awt.event.MouseEvent evt) {
68         colorDibujar=new Color(0,255,255);
69     }
70
71     private void AzulMouseClicked(java.awt.event.MouseEvent evt) {
72         colorDibujar=new Color(0,0,255);
73     }
74 }

```

```

563 private void MoradoMouseClicked(java.awt.event.MouseEvent evt) {
564     colorDibujar=new Color(102,0,102);
565 }
566
567 private void NegroMouseClicked(java.awt.event.MouseEvent evt) {
568     colorDibujar=new Color(0,0,0);
569 }
570
571 // PLACEHOLDERS
572 private void jTextFieldFiguralFocusGained(java.awt.event.FocusEvent evt) {
573     String texto=jTextFieldFigural.getText();
574     if(texto.equals("Nombre figura 1...")) {
575         jTextFieldFigural.setText("");
576         jTextFieldFigural.setForeground(Color.BLACK);
577     }
578 }
579
580 private void jTextFieldFiguralFocusLost(java.awt.event.FocusEvent evt) {
581     String texto=jTextFieldFigural.getText();
582     if(texto.equals("")) {
583         jTextFieldFigural.setText("Nombre figura 1...");
584         jTextFieldFigural.setForeground(new Color(153,153,153));
585     }
586 }
587
588
589 private void jTextFieldFigura2FocusGained(java.awt.event.FocusEvent evt) {
590     String texto=jTextFieldFigura2.getText();
591     if(texto.equals("Nombre figura 2...")) {
592         jTextFieldFigura2.setText("");
593         jTextFieldFigura2.setForeground(Color.BLACK);
594     }
595 }
596
597 private void jTextFieldFigura2FocusLost(java.awt.event.FocusEvent evt) {
598     String texto=jTextFieldFigura2.getText();
599     if(texto.equals("")) {
600         jTextFieldFigura2.setText("Nombre figura 2...");
601         jTextFieldFigura2.setForeground(new Color(153,153,153));
602     }
603 }
604
605 // CAMBIAR EL NOMBRE DE LOS CÍRCULOS AL HACER CLICK
606 private void jLabelMouseClicked(java.awt.event.MouseEvent evt) {
607     String n1=jTextFieldFigural.getText();
608     String n2=jTextFieldFigura2.getText();
609
610     if(!n1.equals("") && !n1.equals("Nombre figura 1...") && !n2.equals("") && !n2.equals("Nombre figura 2..."))
611         jLabelFigural.setText(n1);
612         jLabelFigura2.setText(n2);
613
614         VentanaMouseDrag.repaint();
615         VentanaMouseDrag.revalidate();
616     } else {
617         JOptionPane.showMessageDialog(this, "Rellena los dos campos");
618     }
619 }
620
621 // HOVER DEL BOTÓN
622 private void jLabelMouseClicked(java.awt.event.MouseEvent evt) {
623     BotonInsertarNombre.setBackground(new Color(6, 116, 153));
624 }
625
626 private void jLabelMouseClicked(java.awt.event.MouseEvent evt) {
627     BotonInsertarNombre.setBackground(new Color(0,153,204));
628 }
629
630 // Listeners para poder mover las bolas
631 private void FiguralMousePressed(java.awt.event.MouseEvent evt) {
632     offsetX1=evt.getX();
633     offsetY1=evt.getY();
634 }
635
636 private void FiguralMouseDragged(java.awt.event.MouseEvent evt) {
637     int x=evt.getXOnScreen()-offsetX1-Figural.getWidth();
638     int y=evt.getYOnScreen()-offsetY1-Figural.getHeight();
639     Figural.setLocation(x, y);
640 }
641
642
643 private void Figura2MousePressed(java.awt.event.MouseEvent evt) {
644     offsetX2=evt.getX();
645     offsetY2=evt.getY();
646 }
647
648 private void Figura2MouseDragged(java.awt.event.MouseEvent evt) {
649     int x=evt.getXOnScreen()-offsetX2-Figura2.getWidth();
650     int y=evt.getYOnScreen()-offsetY2-Figura2.getHeight();
651     Figura2.setLocation(x, y);
652 }
653
654
655 /**
656  * @param args the command line arguments
657  */
658 public static void main(String args[]) {
659     /* Set the Nimbus look and feel */
660     /* Look and feel setting code (optional) */
661
662     /* Create and display the form */
663     java.awt.EventQueue.invokeLater(new Runnable() {
664         public void run() {
665             new PantallaPrincipal().setVisible(true);
666         }
667     });
668 }
669

```

FUNCIONAMIENTO

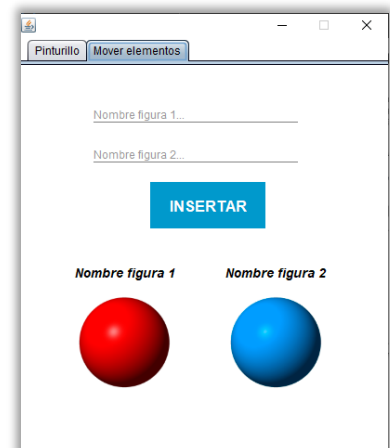
Esta interfaz está compuesta por 2 pestañas, las cuales tienen funciones diferentes. La primera se llama **Pinturillo**, y te permite elegir un color para poder pintar en el área aparece debajo. La segunda tiene 2 figuras, a las cuales les puedes poner un nombre, y mover a lo largo de la interfaz.

EXPLICACIÓN

Los componentes que forman esta interfaz son los siguientes:

- 1 TabbedPane, con 2 JPanel para crear las pestañas.
- 1 JPanel, que actúa como contenedor para los colores. A su vez, este contiene un panel para cada color.
- 1 JPanel para el área de dibujo.
- 2 JTextFields para los nombres de las figuras.
- 1 botón formado por un JPanel y un JLabel.
- 2 JPanels, que sirven como contenedor para cada figura. A su vez, estos contienen JLabel para el nombre y el icono.

Se crea un nuevo borde con la clase **MatteBorder**, que solo dibuja un borde abajo, y se inserta en los JTextFields. También, se crea un color transparente, para poder utilizarlo posteriormente en el color de fondo de los contenedores de las figuras.



Para empezar, creo varias variables, las cuales se usarán para calcular la distancia de X e Y, y para definir el color con el que voy a pintar. La del color la usaré en un Listener de Click en cada JPanel de color. Cada vez que se haga Click en uno, la variable **colorDibujar** tendrá el valor del color del panel que hemos pulsado.

El JPanel **ContenedorPintar** tendrá un Listener de tipo **MouseMove**, mediante el cual, según se mueva el ratón, irá creando cuadrados del mismo color que la variable **colorDibujar**. Para crear estos cuadrados, se hace uso de la clase **Graphics**. El color se define con la función **.setColor** y el tipo de elemento (cuadrado) con **.fillRect**.

Ahora pasamos a la pestaña de **Mover elementos**. En esta, los 2 JTextField tendrán la misma funcionalidad de Placeholder que los JTextField del ejercicio anterior. Se hará mediante los Listener de **FocusGained** y **FocusLost**. Una vez hayamos escrito el texto, pulsaremos en el botón de INSERTAR. Si el texto de estos JTextField es válido, el nombre de las bolas se cambiará. En caso contrario, se lanzará un Dialog.

Los contenedores Figura1 y Figura2 contienen el nombre y la imagen de cada bola. Si mantenemos Click sobre ellos y arrastramos por la pantalla, se podrán mover a lo largo de esta. Esto es posible gracias al Listener **MouseDragged**. Para ello, utilizamos el evento (**evt**) y obtenemos su posición X e Y, además de su posición sobre la pantalla.

Finalmente, para evitar problemas al cambiar el nombre, hago un **.repaint()** y **.revalidate()** del panel completo de esta pestaña Mover elementos. Esto lo hago, ya que, al cambiar el nombre, se dibujan otros elementos debajo del label, a pesar de no

haber programado esta función. Por ello, al cambiar el nombre, se vuelve a dibujar la pestaña entera, y de esta manera, se cambia correctamente el nombre.

BIBLIOGRAFÍA

20,000,000+ vectores de Logo technology, imágenes vectoriales | Depositphotos. (s. f.).
Depositphotos. <https://depositphotos.com/es/vectors/logo-technology.html>