TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE LEÓN

Ing. EN SISTEMAS COMPUTACIONALES

Tópicos avanzados de programación.

Práctica “Salir de un Laberinto”

PRESENTA:

Jonathan de Jesús Pérez Becerra.

CON LA ASESORÍA DE:

Carlos Rafael Levy Rojas.

León, Guanajuato, abril de 2023.

Introducción:

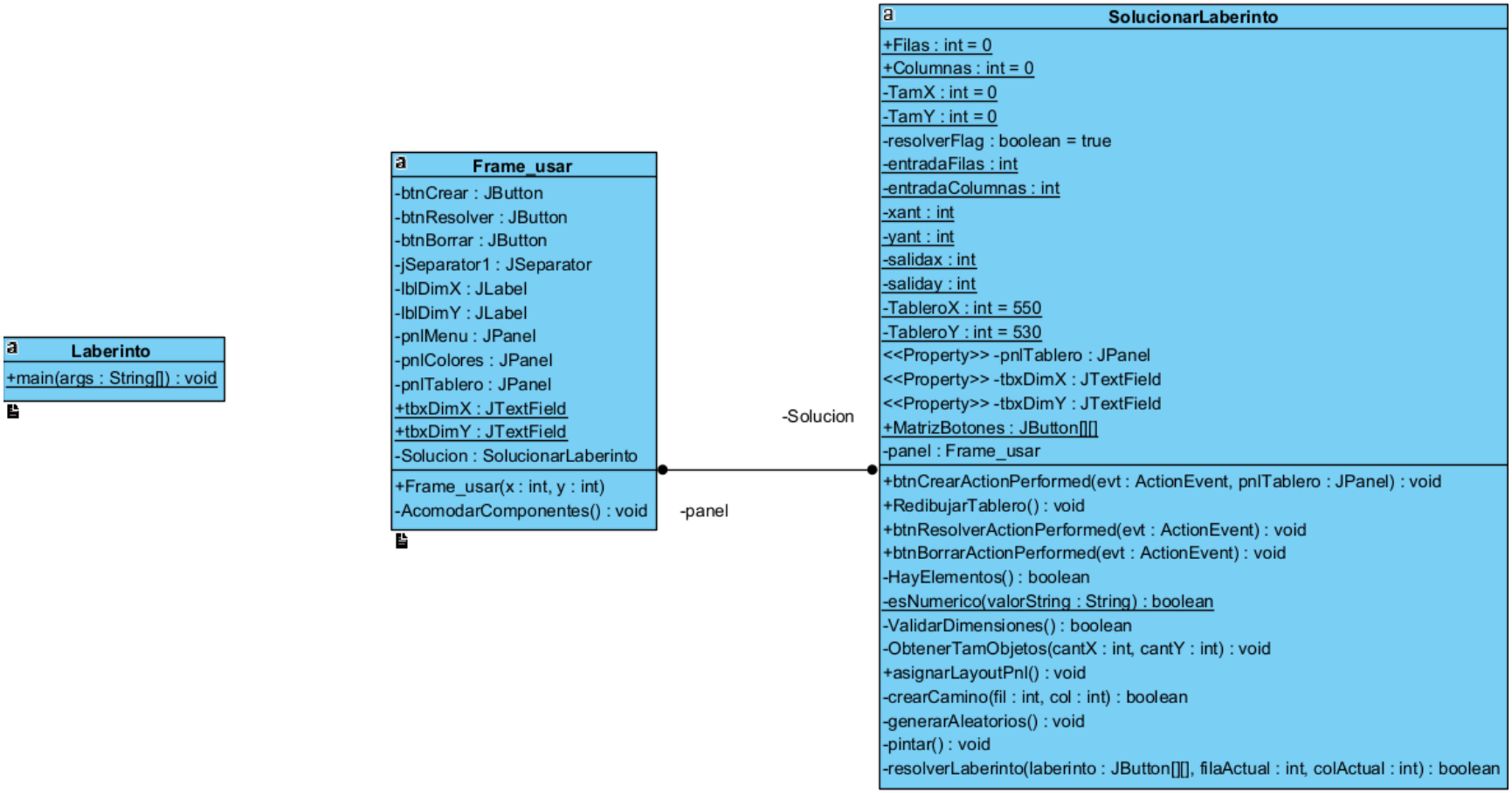
Mi programa que genera y resuelve laberintos en Java es una herramienta interesante y desafiante. A través de la utilización de algoritmos, estructuras de datos y técnicas de programación avanzadas, es capaz de crear un laberinto aleatorio y encontrar el camino correcto a través de él.

La generación de laberintos es una tarea compleja que implica la utilización de algoritmos para determinar las paredes y pasillos del laberinto. Por otro lado, la resolución de laberintos requiere la implementación de algoritmos que permitan encontrar la ruta desde el punto de inicio hasta el punto final.

La combinación de ambos algoritmos en un solo programa es lo que hace que tu programa sea interesante. Permite al usuario generar laberintos aleatorios y resolverlos, lo que puede ser útil para juegos, ejercicios de pensamiento lógico o como un reto para los usuarios.

En definitiva, mi programa de generación y resolución de laberintos en Java es una muestra del poder y la flexibilidad que ofrece este lenguaje de programación, y puede ser utilizado como una herramienta interesante y desafiante para diferentes propósitos.

Diagramas UML



Codigo del main:

```
package laberinto;

import Frame_mostrado.Frame_usar;
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JFrame;

/**
 *
 * @author Jonathan de Jesus Perez Becerra
 */
public class Laberinto {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Frame_usar frame = new Frame_usar(4 ,4);
        frame.setVisible(true);

        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        frame.setLocation(dim.width/2-frame.getSize().width/2, dim.height/2-
frame.getSize().height/2);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

}
```

Clase de la ventana gráfica:

```
package Frame_mostrado;

import Resolucion.SolucionarLaberinto;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSeparator;
import javax.swing.JTextField;
import static javax.swing.LayoutStyle.ComponentPlacement.RELATED;
import static javax.swing.LayoutStyle.ComponentPlacement.UNRELATED;
import static javax.swing.SwingConstants.VERTICAL;
import static javax.swing.BorderFactory.createLineBorder;
import static javax.swing.GroupLayout.Alignment.BASELINE;
import static javax.swing.GroupLayout.Alignment.LEADING;
import static javax.swing.GroupLayout.Alignment.TRAILING;
import static javax.swing.GroupLayout.DEFAULT_SIZE;
import static javax.swing.GroupLayout.PREFERRED_SIZE;

/**
 * Clase que crea un laberinto de forma aleatoria y lo resueve con backtracking
 *
 * @author Jonathan de Jesus Perez Becerra
 */
public class Frame_usar extends JFrame {

    private SolucionarLaberinto Solucion;

    private JButton btnCrear; //Creamos una variable tipo JButton
    private JButton btnResolver; //Creamos una variable tipo JButton
    private JButton btnBorrar; //Creamos una variable tipo JButton
    private JSeparator jSeparator1; //Creamos una variable tipo JSeparator
    private JLabel lblDimX; //Creamos una variable tipo JLabel
    private JLabel lblDimY; //Creamos una variable tipo JLabel
    private JPanel pnlMenu; //Creamos una variable tipo JPanel
    private JPanel pnlColores; //Creamos una variable tipo JPanel
    private JPanel pnlTablero; //Creamos una variable tipo JPanel
    public static JTextField tbxDimX; //Creamos una variable tipo JTextField
    public static JTextField tbxDimY; //Creamos una variable tipo JTextField
```

```
    public Frame_usar(int x, int y) {
        super("Laberinto");
        AcomodarComponentes();
    }

    /**
     * Método que acomoda los Componentes y los coloca en su posicion
     */
    private void AcomodarComponentes() {
        this.Solucion = new Resolucion.SolucionarLaberinto();
        pnlMenu = new JPanel(); //Creamos un Objeto tipo JPanel
        pnlColores = new JPanel(); //Creamos un Objeto tipo JPanel
        pnlTablero = new JPanel(); //Creamos un Objeto tipo JPanel
        lblDimX = new JLabel(); //Creamos un Objeto tipo JLabel
        lblDimY = new JLabel(); //Creamos un Objeto tipo JLabel
        tbxDimX = new JTextField(); //Creamos un Objeto tipo JTexField
        tbxDimY = new JTextField(); //Creamos un Objeto tipo JTextField
        btnResolver = new JButton(); //Creamos un Objeto tipo JButton
        btnCrear = new JButton(); //Creamos un Objeto tipo JButton
        btnBorrar = new JButton(); //Creamos un Objeto tipo JButton
        jSeparator1 = new JSeparator(); //Creamos un Objeto tipo JSeparator

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//Asignamos el sistema
de cierre
        setResizable(false);//Bloqueamos la opcion de hacer grande
        this.setSize(new Dimension(550, 530));//le asignamos la dimencion al Frame

        lblDimX.setText("Elementos en X:");//Asignamos Texto a la primer etiqueta

        tbxDimX.setPreferredSize(new Dimension(50, 24));//Asignamos el tamaño

        lblDimY.setText("Elementos en Y:");//Asignamos Texto a la segunda etiqueta

        tbxDimY.setPreferredSize(new Dimension(50, 24));//Asignamos el tamaño

        btnResolver.setText("Resolver");////Asignamos Texto al boton de Resolver
        btnResolver.setToolTipText("Botón que resuelve el laberinto");//Texto que
aparece si dejas el cursor sobre el boton
        btnResolver.addActionListener(new ActionListener() {//asignamos que el
Botón tendra una accion
            public void actionPerformed(ActionEvent evt) {//creamos el evento
                Solucion.btnResolverActionPerformed(evt);//mandamos llamar el método
que contiene el evento de resolver
            }//cerramos el método
        });//cerramos la accion
```

```
        btnCrear.setText("Crear");
        btnResolver.setToolTipText("Botón que crea el laberinto");//Texto que
aparece s dejas el cursor sobre el boton
        btnCrear.addActionListener(new ActionListener() {//asignamos que el Botón
tendra una accion
            public void actionPerformed(ActionEvent evt) {//creamos el método que
contendra el evento
                Solucion.btnCrearActionPerformed(evt, pnlTablero);//mandamos llamar
el método que contiene el evento de resolver
            }//cerramos el método
        });//cerramos la accion

        btnBorrar.setText("Borrar");
        btnResolver.setToolTipText("Botón que borra los componentes del
laberinto");//Texto que aparece s dejas el cursor sobre el boton
        btnBorrar.addActionListener(new ActionListener() {//asignamos que el Botón
tendra una accion
            public void actionPerformed(ActionEvent evt) {//creamos el método que
contendra el evento
                Solucion.btnBorrarActionPerformed(evt);//mandamos llamar el método
que contiene el evento de resolver
            }//cerramos el método
        });//cerramos la accion

        jSeparator1.setOrientation(VERTICAL);//asignamos que el separador sera
vertical

        GroupLayout pnlMenuLayout = new GroupLayout(pnlMenu);//creamos un
grupo que tendra como host a pnlMenu
        pnlMenu.setLayout(pnlMenuLayout);//asignamos que pnlMenu tendra un
layout tipo GroupLayout
        pnlMenuLayout.setHorizontalGroup(//ponemos que el grupo se guiara de
forma horizontal
            pnlMenuLayout.createParallelGroup(LEADING)//asignamos que los
componentes se alinearan hacia
                    //el origen y de izquierda a derecha
                    .addGroup(pnlMenuLayout.createSequentialGroup()//asignamos
que el grupo se alineara como vaya llegando
                            .addComponent(lblDimX)//añadimos la primer etiiqueta
                            .addPreferredGap(UNRELATED)//asignamos una distancia
para el proximo Bóton
                            .addComponent(tbxDimX, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)//Añadimos el JtextField
                            .addGap(18, 18, 18)//le asignamos las separaciones y el
tamaño
                            .addComponent(lblDimY)//añadimos el JTextField del eje Y
                            .addPreferredGap(UNRELATED)//separamos de la etiqueta
```

```
                        .addComponent(tbxDimY, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)//asignamos las dimensiones
                        .addPreferredGap(RELATED)//marcamos la separacion
                        .addComponent(jSeparator1, PREFERRED_SIZE, 2,
PREFERRED_SIZE)//Añadimos el separador y le asignamos tamaño
                        .addPreferredGap(RELATED)//lo relacionamos
                        .addComponent(btnCrear)//añadimos el boton de crear
                        .addPreferredGap(RELATED)//lo relacionamos para la
separacion
                        .addComponent(btnResolver)//añadimos el boton de resolver
                        .addPreferredGap(RELATED)//lo relacionamos para la
separacion
                        .addComponent(btnBorrar)//añadimos el boton de borrar
                        .addGap(0, 0, Short.MAX_VALUE))//asignamos la separacion
del final
        );
        pnlMenuLayout.setVerticalGroup(//aignamos  el grupo como vertical para
asignar la altura entre todos
                pnlMenuLayout.createParallelGroup(LEADING)//asignamos el menu
paralelo como principal
                        .addGroup(TRAILING,
pnlMenuLayout.createSequentialGroup()//añadimos los componentes como van
añadiendose
                        .addContainerGap(DEFAULT_SIZE,
Short.MAX_VALUE)//asignamos las separasiones automaticas

.addGroup(pnlMenuLayout.createParallelGroup(LEADING)//añadimos el grupo a
principal
                                .addComponent(jSeparator1, PREFERRED_SIZE, 32,
PREFERRED_SIZE)//añadimos el separador

.addGroup(pnlMenuLayout.createParallelGroup(BASELINE)//añadimos el prupo
con un alineamiento base
                                        .addComponent(lblDimX)//añadimos la etiqueta en X
                                        .addComponent(tbxDimX, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)//asignamos su separacion
                                        .addComponent(lblDimY)//añadimos la etiqueta en Y
                                        .addComponent(tbxDimY, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)//asignamos su separacion
                                        .addComponent(btnCrear)//añadomps el boton de
crear
                                        .addComponent(btnResolver)//añadimos el boton de
resolver
                                        .addComponent(btnBorrar))))//añadimos el boton de
borrar
        );
```

```
        pnlTablero.setBackground(new Color(204, 204, 204));//asignamos el color al
panel
        pnlTablero.setBorder(createLineBorder(new Color(0, 0, 0)));//le creamos
bordes negros

//        Solucion.asignarLayoutPnl(this);//Mandamos llamar el metodo de asignal
Layout al panel

        GroupLayout layout = new GroupLayout(getContentPane());//creamos un
objeto tipo GroupLayout
        getContentPane().setLayout(layout);//asignamos el layout al contenido del
panel
        layout.setHorizontalGroup(//asignamos el layout de horizontal
                layout.createParallelGroup(LEADING)//creamos el grupo principal
                        .addGroup(layout.createSequentialGroup()//añadimos el grupo
                                .addContainerGap()//representamos los espacios entre el
borde izquierdo
                                .addGroup(layout.createParallelGroup(LEADING)//creamos
otro grupo como principal
                                        .addGroup(layout.createSequentialGroup()//añadimos los
elementos secuencialmente
                                                .addComponent(pnlTablero, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)//añadimos el tablero
                                                .addGap(0, 0, Short.MAX_VALUE))//asignamos las
separaciones
                                        .addComponent(pnlMenu, DEFAULT_SIZE,
DEFAULT_SIZE, Short.MAX_VALUE))//asignamos los tamaños
                                .addContainerGap())//representamos los espacios entre el
borde derecho
        );
        layout.setVerticalGroup(//ahora lo hacemos de vertical
                layout.createParallelGroup(LEADING)//creamos el grupo principal
                        .addGroup(layout.createSequentialGroup()//añadimos el grupo de
forma secuencial
                                .addContainerGap()//asignamos el espacio entre el borde
superior
                                .addComponent(pnlMenu, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)//añadimos el menu
                                .addPreferredGap(RELATED)//creamos la separacion del
panel con el menu en 0
                                .addComponent(pnlTablero, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)//añadimos el tablero
                                .addContainerGap(0, Short.MAX_VALUE))//asignamos la
separacion del borde inferior
        );
```

```
        pnlColores.setLayout(new GridLayout(0, 4)); // 0 filas y 8 columnas para
acomodar los colores y etiquetas

        //añadimos un nuevo panel y lo pintamos
        pnlColores.add(new JPanel() {
            {
                setBackground(Color.blue);
                JLabel entrada = new JLabel("Entrada");
                add(entrada);
                entrada.setForeground(Color.white);
            }
        });

        //añadimos un nuevo panel y lo pintamos
        pnlColores.add(new JPanel() {
            {
                setBackground(Color.red);
                JLabel salida = new JLabel("Salida");
                add(salida);
                salida.setForeground(Color.white);
            }
        });

        //añadimos un nuevo panel y lo pintamos
        pnlColores.add(new JPanel() {
            {
                setBackground(Color.green);
                JLabel camino = new JLabel("Camino");
                camino.setForeground(Color.white);
                add(camino);
            }
        });

        //añadimos un nuevo panel y lo pintamos
        pnlColores.add(new JPanel() {
            {
                setBackground(Color.cyan);
                JLabel BackTracking = new JLabel("BackTracking");
                BackTracking.setForeground(Color.white);
                add(BackTracking);
            }
        });

        // Definición del layout general del formulario
        layout.setHorizontalGroup(
                layout.createParallelGroup(LEADING)
                        .addGroup(layout.createSequentialGroup()
```

```
                        .addContainerGap()
                        .addGroup(layout.createParallelGroup(LEADING)
                            .addComponent(pnlTablero, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)
                            .addComponent(pnlMenu, DEFAULT_SIZE,
DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(pnlColores, DEFAULT_SIZE,
DEFAULT_SIZE, Short.MAX_VALUE)
                        )
                        .addContainerGap()
                )
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(pnlMenu, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)
                    .addPreferredGap(RELATED)
                    .addComponent(pnlTablero, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)
                    .addPreferredGap(RELATED)
                    .addComponent(pnlColores, PREFERRED_SIZE,
DEFAULT_SIZE, PREFERRED_SIZE)
                    .addContainerGap(0, Short.MAX_VALUE))
        );
        // Definición del layout del panel de tablero
        GroupLayout pnlTableroLayout = new GroupLayout(pnlTablero);
        pnlTablero.setLayout(pnlTableroLayout);
        pnlTableroLayout.setHorizontalGroup(
            pnlTableroLayout.createParallelGroup(LEADING)
                .addGap(0, 585, Short.MAX_VALUE)
        );
        pnlTableroLayout.setVerticalGroup(
            pnlTableroLayout.createParallelGroup(LEADING)
                .addGap(0, 500, Short.MAX_VALUE)
        );
        Solucion.setPnlTablero(pnlTablero);//mandamos el tablero a Solucionar
laberinto
        Solucion.setTbxDimX(tbxDimX);//mandamos el cuadro de texto de X a
Solucionar laberinto
        Solucion.setTbxDimY(tbxDimY);//mandamos el cuadro de texto de Y a
Solucionar laberinto

        pack();//asignamos el tamaño de los componentes por si solos
    }
}
```

Clase de funcionamiento:

```
package Resolucion;

import java.awt.Color;
import java.awt.GridLayout;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.util.Random;
import java.util.Stack;
import javax.swing.GroupLayout;
import static javax.swing.GroupLayout.Alignment.LEADING;
import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 *
 * @author Jonathan de Jesus Perez Becerra
 */
public class SolucionarLaberinto {

    //Variables globales del formulario
    public static int Filas = 0;
    public static int Columnas = 0;
    private static int TamX = 0;
    private static int TamY = 0;
    private boolean resolverFlag = true;
    private static int entradaFilas, entradaColumnas, xant, yant, salidax, saliday;
    private static final int TableroX = 550;
    private static final int TableroY = 530;
    private JPanel pnlTablero;
    private JTextField tbxDimX;
    private JTextField tbxDimY;

    //Matriz de botones
    public static JButton[][] MatrizBotones;

    private Frame_mostrado.Frame_usar panel;//instanciamos la clase

    public void btnCrearActionPerformed(ActionEvent evt, JPanel pnlTablero) {
        if (HayElementos()) {
            JOptionPane.showMessageDialog(null, "Primero debes borrar el laberinto
actual!");
        } //Fin if HayElementos
        else {
```

```
            //Se verifica que las entradas sean validas
            if (ValidarDimensiones()) {
                //Se settea el tamaño de la matriz de botones
                MatrizBotones = new JButton[Filas][Columnas];
                //Se settea el tamaño de gridLayout de nuestro panel del tablero
                pnlTablero.setLayout(new GridLayout(Filas, Columnas));
                //Se obtiene el tamaño de los botones acorde a su cantidad
                ObtenerTamObjetos(Filas, Columnas);
                //Se declaran los contadores a utilizar
                int entradaAleatoria;
                entradaAleatoria = (int) (Math.random() * 4);
                if (entradaAleatoria == 0) {
                    entradaFilas = 0;
                    entradaColumnas = (int) (Math.random() * Columnas - 1);
                    if (entradaColumnas == 0) {
                        entradaColumnas = 1;
                    }
                }
                if (entradaAleatoria == 1) {
                    entradaColumnas = Columnas - 1;
                    entradaFilas = (int) (Math.random() * Filas - 1);
                    if (entradaFilas == 0) {
                        entradaFilas = 1;
                    }
                }
                if (entradaAleatoria == 2) {
                    entradaFilas = Filas - 1;
                    entradaColumnas = (int) (Math.random() * Columnas - 1);
                    if (entradaColumnas == 0) {
                        entradaColumnas = 1;
                    }
                }
                if (entradaAleatoria == 3) {
                    entradaColumnas = 0;
                    entradaFilas = (int) (Math.random() * Filas - 1);
                    if (entradaFilas == 0) {
                        entradaFilas = 1;
                    }
                }

                //Creamos un algoritmo que indique la entrada
                for (int filas = 0; filas < Filas; filas++) {//Se recorre la dimension X desde
0 hasta DimensionX
                    for (int columnas = 0; columnas < Columnas; columnas++) {//Se
recorre la dimension Y desde 0 hasta DimensionY
```

```
                    JButton btnNuevo = new JButton();//Se crea un nuevo objeto de
tipo JButton
                    if ((columnas == 0 | columnas == Columnas - 1) | (filas == 0 | filas
== Filas - 1)) {
                        btnNuevo.setText(1 + "");//le asignamos un 1 para simular un
muro
                    } else {
                        btnNuevo.setText("");
                    }
                    btnNuevo.setSize(TamX, TamY);//Se le asignan sus dimensiones
(ancho, alto)
                    if (filas == entradaFilas && columnas == entradaColumnas) {//Si
estamos en la posicion de entrada
                        btnNuevo.setBackground(Color.blue);//Coloreamos la entrada de
AZUL
                        btnNuevo.setText("*");
                    }
                    MatrizBotones[filas][columnas] = btnNuevo;//Se agrega a la matriz
el botón recien creado
                    pnlTablero.add(MatrizBotones[filas][columnas]);//Se agrega al panel
                    RedibujarTablero();//Se redibuja el panel
                }//Fin For - Y
            }//Fin For - X
            crearCamino(entradaFilas, entradaColumnas);
            generarAleatorios();
            pintar();

        }//Fin If - ValidaDimensiones
        else {
            JOptionPane.showMessageDialog(null, "Las Dimensiones a ingresar
deben ser númericas y en un rango de entre 3 y 50");
        }//Fin if - ValidaDimensiones - false
    }
}

/**
 * Metodo que redibuja el elemto pnlTablero
 */
public void RedibujarTablero() {
    //Se valida los componentes del elemento pnlTablero
    pnlTablero.validate();
    //Se redibuja el elemento pnlTablero y sus componentes hijos
    pnlTablero.repaint();
}

/**
 * Metodo que resuelve el Laberinto
```

```
     *
     * @param evt evento que viene del clic del boton Resolver
     */
    public void btnResolverActionPerformed(ActionEvent evt) {
        //Se valida que hayan elementos en el panel
        if (HayElementos()) {
            if (resolverFlag) {
                resolverLaberinto(MatrizBotones, entradaFilas, entradaColumnas);
                resolverFlag = false;
            } else {
                JOptionPane.showMessageDialog(null, "El laberinto ya fue resuelto!");
            }
        }//Fin if HayElementos
        else {
            JOptionPane.showMessageDialog(null, "Primero debes crear el
laberinto!");
        }
    }

    /**
     * Metodo que borra los componentes del panel
     *
     * @param evt evento que viene del clic del boton Borrar
     */
    public void btnBorrarActionPerformed(ActionEvent evt) {
        if (HayElementos()) {
            //Se recorre la matriz de botones y se elimina cada elemento de la matríz
            for (int x = 0; x < Filas; x++) {
                for (int y = 0; y < Columnas; y++) {
                    MatrizBotones[x][y] = null;
                }//Fin For - y
            }//Fin For - x
            //Se remueven todos los elementos hijos del JPanel pnlTablero
            pnlTablero.removeAll();
            //Se redibuja el panel
            RedibujarTablero();
            resolverFlag = true;
        } //Fin if HayElementos
        else {
            JOptionPane.showMessageDialog(null, "Primero Debes crear el
laberinto!");
        }
    }

    /**
     * Función que verifica si hay elementos creados en el panel
     */
```

```
    private boolean HayElementos() {
        //Si hay elementos en el panel retorna true, caso contrario retorna false
        return pnlTablero.getComponentCount() > 0;
    }

    /**
     * Función que verifica si el valor String es de tipo númerico
     */
    private static boolean esNumerico(String valorString) {
        try {
            Integer.parseInt(valorString);
        } catch (NumberFormatException ex) {
            return false;
        }
        return true;
    }

    /**
     * Función que valida que las dimensiones ingresadas por el usuario sean
     * validas y entre los rangos de 3 a 50
     */
    private boolean ValidarDimensiones() {
        boolean valido = false;//definición inicial de variable de retorno
        if (esNumerico(tbxDimX.getText())) {//Se obtienen las entradas de texto de las
dimensiones ingresadas
            if (esNumerico(tbxDimY.getText())) {
                Filas = Integer.parseInt(tbxDimX.getText());//Se obtiene el tamaño en X
(ancho) que debe tener el boton
                Columnas = Integer.parseInt(tbxDimY.getText());//Se obtiene el tamaño
en Y (alto) que debe tener el boton
                if (((Filas < 51) && (Columnas < 51)) && ((Filas > 2) && (Columnas > 2)))
{//Se verifican que las dimensiones ingresadas por el usuario esten entre 1 y 50
                    return true;
                }
            }
        }
        return valido;
    }

    /**
     * Metodo que calcula el tamaño de ancho y alto de los botones acorde a la
     * cantidad de elementos en la matriz
     */
    private void ObtenerTamObjetos(int cantX, int cantY) {
        TamX = TableroX / cantX;
        TamY = TableroY / cantY;
    }
```

```
    /**
     * Metodo que asigna el Layout al panel
     */
    public void asignarLayoutPnl() {
        // Se crea un nuevo objeto GroupLayout que se utilizará para establecer la
disposición del panel pnlTablero
        GroupLayout pnlTableroLayout = new GroupLayout(pnlTablero);

        // Se establece la disposición horizontal del panel pnlTablero
        pnlTablero.setLayout(pnlTableroLayout);
        pnlTableroLayout.setHorizontalGroup(

pnlTableroLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                        .addGap(0, 585, Short.MAX_VALUE)
        );

        // Se establece la disposición vertical del panel pnlTablero
        pnlTableroLayout.setVerticalGroup(

pnlTableroLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                        .addGap(0, 500, Short.MAX_VALUE)
        );
    }

    /**
     * Método que crea un camino entre la entrada y la salida para que siempre
     * exista solucion
     *
     * @param fil valor de la entrada en x
     * @param col valor de la entrada en y
     * @return utilizamos la llamada recursiva o true en caso de haber acabado
     * el camino
     */
    private boolean crearCamino(int fil, int col) {
        asignarLayoutPnl();
        //si la entrada esta a la izquierda
        if (col == 0) {
            MatrizBotones[fil][col + 1].setText("*");
            MatrizBotones[fil][col + 1].setBackground(Color.cyan);
            return crearCamino(fil, col + 1);
        }
        //si la entrada esta a la derecha
        if (col == Columnas - 1) {
            MatrizBotones[fil][col - 1].setText("*");
            MatrizBotones[fil][col - 1].setBackground(Color.cyan);
            return crearCamino(fil, col - 1);
```

```
        }
        //si la entrada esta arriba
        if (fil == 0) {
            MatrizBotones[fil + 1][col].setText("*");
            MatrizBotones[fil + 1][col].setBackground(Color.cyan);
            return crearCamino(fil + 1, col);
        }
        //si la estrada esta abajo
        if (fil == Filas - 1) {
            MatrizBotones[fil - 1][col].setText("*");
            MatrizBotones[fil - 1][col].setBackground(Color.cyan);
            return crearCamino(fil - 1, col);
        }

        //---------------------------------------------------------------------
        Random aleatorio = new Random();
        int eleccion = aleatorio.nextInt(4);

        //generamos el camino hacia arriba
        if (eleccion == 0 && !MatrizBotones[fil - 1][col].getText().equals("*") && fil - 1
!= xant && fil - 1 != entradaFilas) {//arriba
            if (MatrizBotones[fil - 1][col].getText().equals(1 + "")) {
                MatrizBotones[fil - 1][col].setText("*");
                MatrizBotones[fil - 1][col].setBackground(Color.red);
                salidax = fil - 1;
                saliday = col;
                return true;
            } else {
                xant = fil;
                yant = col;
                MatrizBotones[fil - 1][col].setText("*");
                return crearCamino(fil - 1, col);
            }
        }
        //generamos el camino hacia la derecha
        if (eleccion == 1 && !MatrizBotones[fil][col + 1].getText().equals("*") && col + 1
!= yant && col + 1 != entradaColumnas) {//derecha
            if (MatrizBotones[fil][col + 1].getText().equals(1 + "")) {
                MatrizBotones[fil][col + 1].setText("*");
                MatrizBotones[fil][col + 1].setBackground(Color.red);
                salidax = fil;
                saliday = col + 1;
                return true;
            } else {
                xant = fil;
                yant = col;
                MatrizBotones[fil][col + 1].setText("*");
```

```
                return crearCamino(fil, col + 1);
            }
        }
        //generamos el camino hacia abajo
        if (eleccion == 2 && !MatrizBotones[fil + 1][col].getText().equals("*") && fil + 1
!= xant && fil + 1 != entradaFilas) {//abajo
            if (MatrizBotones[fil + 1][col].getText().equals(1 + "")) {
                MatrizBotones[fil + 1][col].setText("*");
                MatrizBotones[fil + 1][col].setBackground(Color.red);
                salidax = fil + 1;
                saliday = col;
                return true;
            } else {
                xant = fil;
                yant = col;
                MatrizBotones[fil + 1][col].setText("*");
                return crearCamino(fil + 1, col);
            }
        }
        //generamos el camino hacia la izquieda
        if (eleccion == 3 && !MatrizBotones[fil][col - 1].getText().equals("*") && col - 1
!= yant && col - 1 != entradaColumnas) {//izquierda
            if (MatrizBotones[fil][col - 1].getText().equals(1 + "")) {
                MatrizBotones[fil][col - 1].setText("*");
                MatrizBotones[fil][col - 1].setBackground(Color.red);
                salidax = fil;
                saliday = col - 1;
                return true;
            } else {
                xant = fil;
                yant = col;
                MatrizBotones[fil][col - 1].setText("*");
                return crearCamino(fil, col);
            }
        }

        return crearCamino(fil, col);
    }

    /**
     * Metodo que genera numeros aleatorios entre 0 y 1 respetando el camino
     * creado anteriormente
     */
    private void generarAleatorios() {
        /**
         * Recorre la matriz de botones y asigna valores aleatorios de 0 o 1 a
         * los botones vacíos y un valor de 0 a los botones con asterisco.
```

```
         */
        for (int i = 0; i < MatrizBotones.length; i++) {
            for (int j = 0; j < MatrizBotones[0].length; j++) {
                // Verificar si el botón está vacío
                if (MatrizBotones[i][j].getText().equals("")) {
                    // Asignar un valor aleatorio de 0 o 1 al botón vacío
                    MatrizBotones[i][j].setText((int) (Math.random() * 2) + "");
                }
                // Verificar si el botón es la entrada del laberinto
                if (MatrizBotones[i][j].getText().equals("*")) {
                    // Asignar un valor de 0 al botón con asterisco
                    MatrizBotones[i][j].setText(0 + "");
                }
            }
        }
    }

    /**
     * Método que pinta las casillas para que sea mas facil visualizar los
     * caminos de las paredes, asi mismo distinguir la entrada de la salida
     */
    private void pintar() {
        for (int i = 0; i < MatrizBotones.length; i++) {
            for (int j = 0; j < MatrizBotones[0].length; j++) {
                if (MatrizBotones[i][j].getText().equals("1")) {
                    MatrizBotones[i][j].setBackground(Color.lightGray);
                }
                if (MatrizBotones[i][j].getText().equals("0")) {
                    MatrizBotones[i][j].setBackground(Color.white);
                }
                if (i == entradaFilas && j == entradaColumnas) {
                    MatrizBotones[i][j].setBackground(Color.blue);
                    MatrizBotones[i][j].setText("E");
                }
                if (i == salidax && j == saliday) {
                    MatrizBotones[i][j].setBackground(Color.red);
                    MatrizBotones[i][j].setText(0 + "");
                }

            }
        }
    }

    private boolean resolverLaberinto(JButton[][] laberinto, int filaActual, int
colActual) {
        Stack<int[]> camino = new Stack<>(); // Pila para almacenar las coordenadas
del camino actual
```

```
        camino.push(new int[]{filaActual, colActual});

        // Chequear si estamos en el objetivo
        if (laberinto[filaActual][colActual].getText().equals(0 + "") && (colActual == 0 |
colActual == Columnas - 1
                | filaActual == 0 | filaActual == Filas - 1)) {
            JOptionPane.showMessageDialog(null, "Logre salir!");
            return true;
        }

        // Chequear si la casilla actual es una pared o ya ha sido visitada
        if (laberinto[filaActual][colActual].getText().equals(1 + "") ||
laberinto[filaActual][colActual].getText().equals("")) {
            camino.pop(); // Desapilar las coordenadas de la casilla actual
            return false;
        }

        // Marcar la casilla actual como visitada
        if (!laberinto[filaActual][colActual].getText().equals("E")) {
            laberinto[filaActual][colActual].setText("");
            laberinto[filaActual][colActual].setBackground(Color.green);
        }

        // Chequear las casillas vecinas
        if (filaActual > 0 && resolverLaberinto(laberinto, filaActual - 1, colActual)) { //
arriba
            return true;
        }
        if (colActual < laberinto[0].length - 1 && resolverLaberinto(laberinto, filaActual,
colActual + 1)) { // derecha
            return true;
        }
        if (filaActual < laberinto.length - 1 && resolverLaberinto(laberinto, filaActual +
1, colActual)) { // abajo
            return true;
        }
        if (colActual > 0 && resolverLaberinto(laberinto, filaActual, colActual - 1)) { //
izquierda
            return true;
        }

        // Si ninguna casilla vecina lleva al objetivo, desmarcar la casilla actual y
retroceder
        if (!laberinto[filaActual][colActual].getText().equals("E")) {
            laberinto[filaActual][colActual].setText(0 + "");
            laberinto[filaActual][colActual].setBackground(Color.CYAN);
            camino.pop(); // Desapilar las coordenadas de la casilla actual
```

```
        }
        return false;
    }


    public void setPnlTablero(JPanel pnlTablero) {
        this.pnlTablero = pnlTablero;
    }

    public void setTbxDimX(JTextField tbxDimX) {
        this.tbxDimX = tbxDimX;
    }

    public void setTbxDimY(JTextField tbxDimY) {
        this.tbxDimY = tbxDimY;
    }

}
```

Conclusión:

En conclusión, la creación de un programa en Java que genera y resuelve laberintos es una tarea desafiante que requiere una comprensión sólida de los algoritmos y estructuras de datos. A través de la combinación de técnicas avanzadas de programación, es posible crear un programa interesante y útil que puede ser utilizado para diferentes propósitos, como juegos o ejercicios de pensamiento lógico.

Es importante tener en cuenta que la documentación del programa es esencial para facilitar la comprensión y el mantenimiento del código. La documentación debe proporcionar información clara y detallada sobre el propósito del programa, los algoritmos utilizados, los requisitos previos y cómo utilizar el programa.

En resumen, la creación de un programa que genera y resuelve laberintos en Java es un proyecto emocionante que permite explorar las posibilidades y capacidades de este lenguaje de programación. Al mismo tiempo, es importante tener en cuenta la documentación y la planificación cuidadosa para garantizar un código claro y fácilmente mantenible.