UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

ALUMNO:

PAZ MALDONADO CARLOS SAÚL

NOMBRE DEL PROFESOR:

HERNANDEZ CABRERA JESUS

NOMBRE DE LA MATERIA:

ESTRUCTURA DE DATOS

FECHA DE ENTREGA:

12 de octubre del 2024

TAREA: Punto extra. Laberinto

INSTRUCCIONES DE TAREA

Implementar un algoritmo que resuelva laberintos.

Debe emplear los ADT Array2D, Pila(Stack) y el algoritmo de back tracking.

Debe cumplir con los siguientes criterios.

- El orden para buscar un pasillo disponible para moverse es: Izquierda, arriba, derecha y abajo.
- La posición de cada movimiento se mete (push) a la pila.
- Cuando se determine que en un punto en los pasillos no tiene más opciones de búsqueda, se deben marcar para que no sean tomados en cuenta.
- Debe existir un mecanismo simple para configurar el laberinto inicial. El tamaño mínimo del laberinto debe ser de 5 x 5.
- Debe estar marcado el punto de entrada.
- Debe estar marcada el punto de salida.
- Los pasillos y paredes los puedes representar como gustes.

DOCUMENTACIÓN DEL LABERINTO

Estructura del proyecto

Para la implementación de este proyecto se utiliza el patrón de diseño MVC, ya que permite separar las clases en función del papel que desempeña cada una de ellas en el proyecto. Permitiendo la separación de la lógica, el modelo y las vistas.

El paquete **modelos** alberga diferentes tipos de clases divididos en más paquetes. Menciono la composición y lo que alberga cada uno de ellos:

1. Adts: Este paquete alberga todas las adts que se necesitan para construir el laberinto.
    a. ADTArray2D: permite crear una cuadricula de 2 dimensiones de cualquier tipo de dato. Con sus métodos correspondientes para poder manipular este tipo de objeto.
    b. ADTStack: permite almacenar el camino que tendrá que recorrer el laberinto. es igual que una estructura de datos pila, con sus mismos métodos.
    c. ColaADT: es lo mismo que una estructura de datos de cola, el cual, se implementó en algunas ocasiones.
    d. ListaDoblementeLigada: es la base de las demás ADT como la cola o la pila.
    e. NodoDoble: es la base de la lista doblemente ligada.
2. Excepciones
    a. Tiene una clase que me permite crear excepciones. En este caso una excepción para la lectura de archivos FXML.
3. JavaFx
    a. En este paquete se tienen las clases que permiten crear los componentes visuales necesarios para crear el laberinto.
    b. CuadriculaFX: es la cuadricula que en cada celda de esta cuadricula contiene un pane.
    c. Escena: tiene los métodos necesarios para ejecutar un componente visual de javaFX.
    d. MenuFX: es el principal componente que contendrá la cuadriculaFX y los botones necesarios para poder programar la lógica.
4. Laberinto
    a. Coordenada: esta clase permite guardar la ubicación de cada celda, guardando la fila y columna donde esta, igualmente implementa un

valor booleano que permite saber la diferencia entre si es un camino o una pared.

5. LaberintoLogica (Clase)
    a. Esta clase es la lógica del backtracking que posteriormente explicare de forma más detallada.

En el paquete **controladores** tiene esta estructura:

1. Vistas: este paquete almacena los controladores de las vistas que se encuentran en el paquete de vistas.
    a. CeldaController.
    b. CuadriculaController.
    c. MenuController.
2. Laberinto (Clase)
    a. Es el controlador principal que conecta la lógica con la vista.
    b. En esta misma clase se manejan las vistas, los botones, el pintado de las celdas, entre otros métodos que son necesarios para que el laberinto funcione adecuadamente.

En el paquete vistas la estructura se compone de los archivos FXML y un paquete que tiene una clase llamada Vista, creando dos instancias de MenuFX una para el menú principal y otra para cargar el laberinto.

Implementación del backtracking

Para la implementación del backtracking se necesitó implementar dos estructuras de datos la primera de ellas es la pila y la segunda un set.

La pila permite guardar el camino correcto por el que debe tomar el laberinto para llegar a la salida, por otro lado, la estructura de datos set almacena todos los movimientos sin repetirse la misma coordenada.

Para obtener este mecanismo de backtracking se realiza lo siguiente:

A partir de una coordenada especificada, se crean más objetos de la clase coordenada, cuatro en total, que son las vecinas, almacenándolos en una ColaADT temporalmente y en el orden especificado: izquierda, arriba, derecha y abajo. Esta cola entra en un bucle while, que solo sale si esta cola esta vacía. A partir de ello, se desencola cada coordenada en el orden en que fueron insertados (FIFO) por lo que, permite evaluar cual es la siguiente coordenada para moverse. Pero esta coordenada al momento de desencolarse tiene que pasar por ciertas funciones evaluadoras; determinando si es candidata para pertenecer a la pila que almacena el camino.

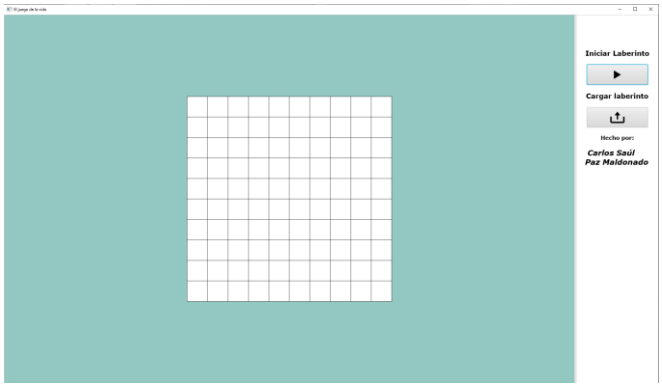
La primera evaluación es determinar si esta en el rango existente de la grid, previniendo que no haya coordenadas menores al rango de cero y mayores al rango existente de la grid, ya sea en filas o en columnas.

La segunda evaluación es determinar si esta coordenada ya ha sido visitada anteriormente, apoyándose de la estructura de datos Set y su método contains. Si no ha sido visitada anteriormente y ha pasado la primera evaluación tiene otro punto a favor para ser considerada una candidata.

Para finalizar, la última evaluación es determinar si este objeto de la clase Coordenada es una pared o un camino. Apoyándose de la misma clase, se obtiene el estado de esa coordenada, si es falso, entonces es una pared; caso opuesto un camino habilitado.

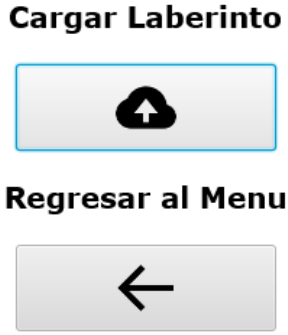
FUNCIONAMIENTO DEL LABERINTO

1. Para ejecutar el laberinto es necesario inicializarlo desde la clase Main.
2. Se encuentra una instancia de la clase Laberinto el cual se podrán modificar el numero de filas y el numero de columnas que requiera.
3. También podrá ingresar la velocidad con la que se pintará el laberinto, donde el numero 1 significará un segundo y también podrá aceptar fracciones de segundo (Ejemplo: 0.100;0.20;0.5…). Si no agrega un numero en el constructor, por defecto tendrá el valor de 1 segundo.
4. Al ejecutar el programa podrá observar que aparece una ventana llamada menú de inicio.

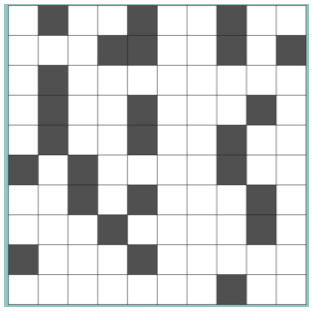


5. Hay dos botones, uno permite arrancar el laberinto y el otro permite cargarlo. Si no ha cargado el laberinto no lo dejara iniciar.

6. Al presionar el botón de cargar laberinto podrá notar que se cambio el menú y el nombre de los botones habrá cambiado.

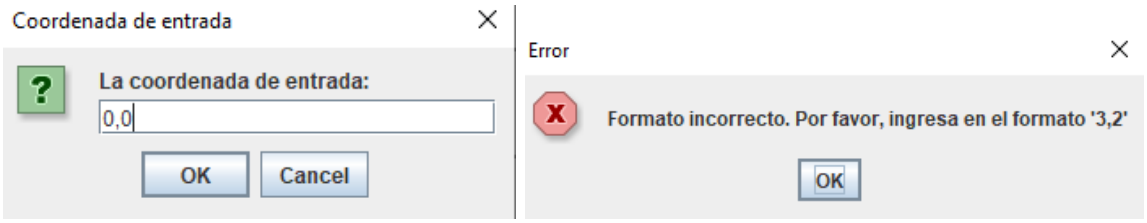


7. Ahora tendrá que configurar su laberinto en este menú. Para cargar las paredes tendrá que dar click en las celdas, este se cambiará a color negro indicando que es una pared, el color blanco indica el camino disponible.

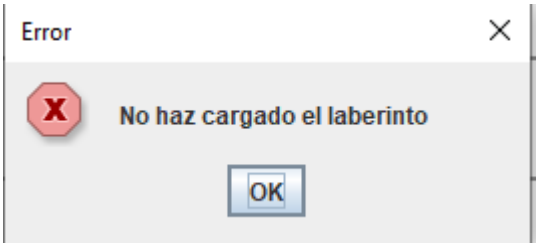


8. Para confirmar los cambios tendrá que dar click en el botón cargar laberinto.

9. Le aparecerán dos ventanas. Tendrá que ingresar las coordenadas de entrada y de salida en el siguiente formato “2,3” (sin comillas). En caso de ingresar mal las coordenadas le aparecerá otro cuadro de dialogo indicando que lo hizo mal y podrá volver a ingresar las coordenadas.

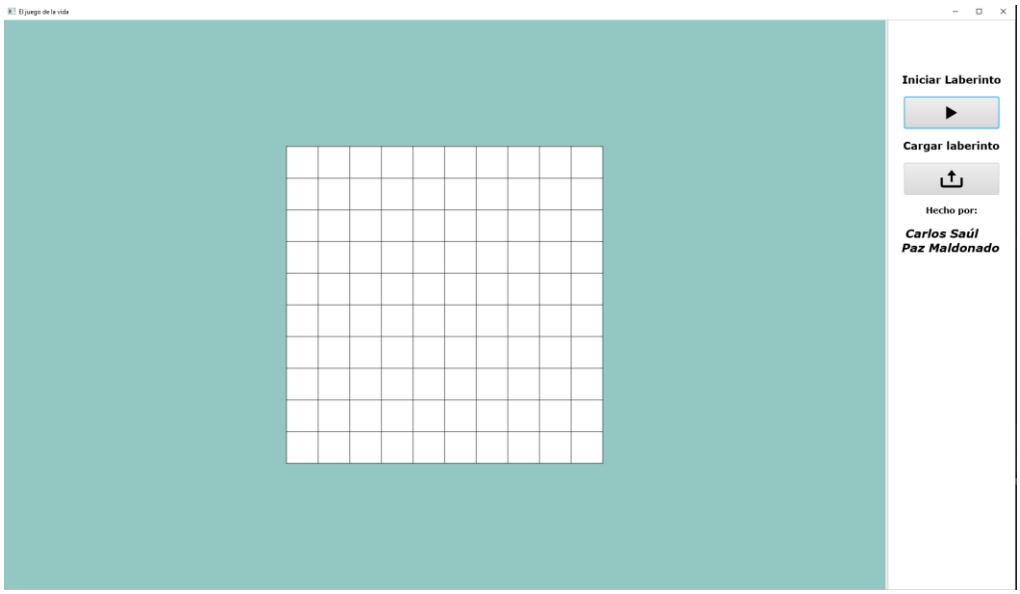


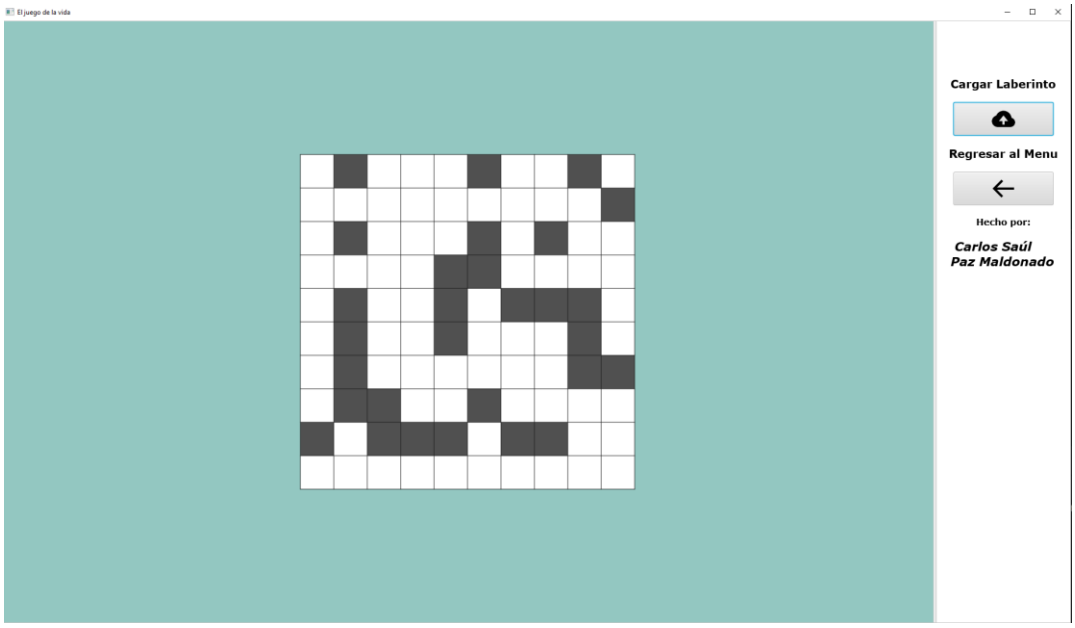
10. Lo regresará automáticamente al menú principal, por lo que podrá iniciar el laberinto dando click en el botón correspondiente. En caso de no haber cargado el laberinto no le permitirá iniciarlo.



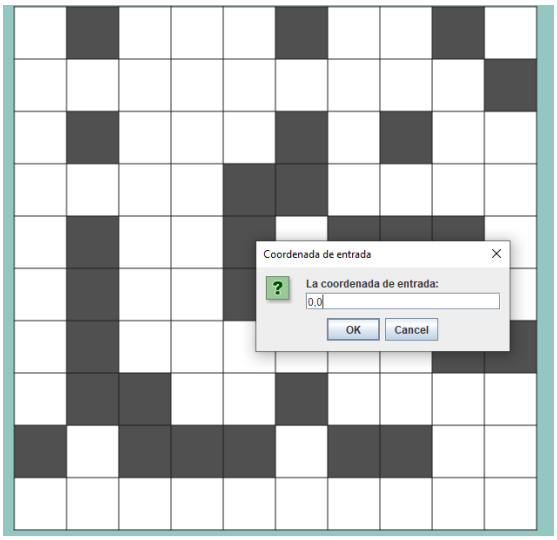
11. Si quiere modificar nuevamente el laberinto podrá hacerlo repitiendo los pasos desde el punto 6.

CAPTURAS DE EJECUCIÓN DEL PROGRAMA

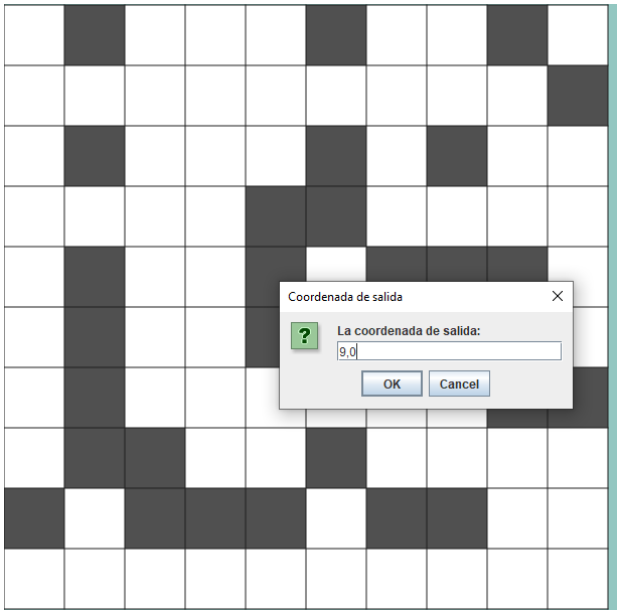
1. Menu de inicio
   a. 

2. Menu para cargar el laberinto
   a. 

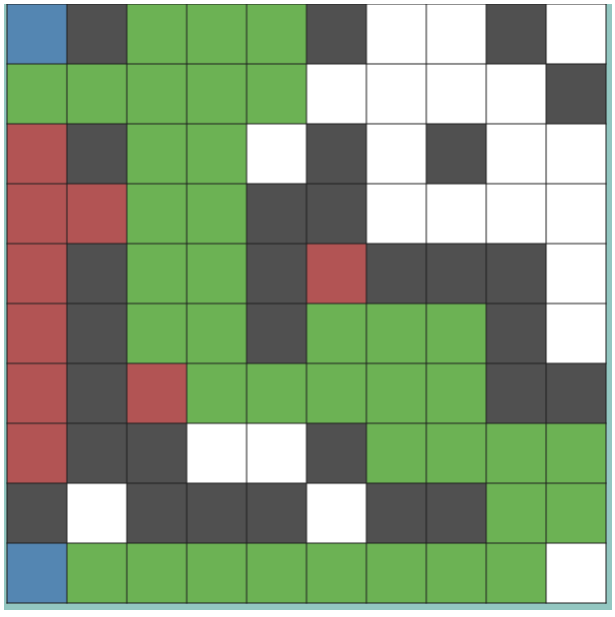
3. Configurando la coordenada de entrada

a. 

4. Configurando la coordenada de salida

a. 

5. Iniciando el laberinto

a. 

CÓDIGOS

1. Clase CeldaController

```
package unam.fesaragon.estructuradatos.controladores.vistas;

import javafx.fxml.FXML;
import javafx.scene.layout.Pane;
import unam.fesaragon.estructuradatos.modelos.laberinto.Coordenada;

public class CeldaController {
    private Coordenada coordenada;
    private static final String colorTrue = "#FFF";
    private static final String colorPared = "#505050";
    public static final String urlFXMLDeCelda =
"/unam/fesaragon/estructuradatos/vistas/celda.fxml";
    private boolean cambiarEstadoDeLasCeldas_Click = true;
    @FXML
    private Pane panelCelda;

    @FXML
    private void onClicked() {
        if (!cambiarEstadoDeLasCeldas_Click) return;
        getCoordenada().setEstado(!getCoordenada().isEstado());
        cambiarColor();
    }

    private void cambiarColor() {

        if (getCoordenada().isEstado()) {
            panelCelda.setStyle("-fx-background-color: " + colorTrue +
";");
        } else {
            panelCelda.setStyle("-fx-background-color: " + colorPared +
";");
        }
    }


    //GETTERS Y SETTERS
    public Pane getPanelCelda() {
        return panelCelda;
    }

    public Coordenada getCoordenada() {
        return coordenada;
    }

    public void setCoordenada(Coordenada coordenada) {
        this.coordenada = coordenada;
```

```
    }

    public boolean isCambiarEstadoDeLasCeldas_Click() {
        return cambiarEstadoDeLasCeldas_Click;
    }

    public void setCambiarEstadoDeLasCeldas_Click(boolean
cambiarEstadoDeLasCeldas_Click) {
        this.cambiarEstadoDeLasCeldas_Click =
cambiarEstadoDeLasCeldas_Click;
    }
}
```

2. Clase CuadriculaController

```
package unam.fesaragon.estructuradatos.controladores.vistas;

import javafx.fxml.FXML;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.Pane;

public class CuadriculaController {
    public static final String urlFXMLDeCuadricula =
"/unam/fesaragon/estructuradatos/vistas/cuadricula.fxml";
    @FXML
    private GridPane gridPaneCuadricula;


    public GridPane getGridPaneCuadricula() {
        return gridPaneCuadricula;
    }
    public CeldaController getCelda(int fila, int columna) {
        Pane panelCelda = (Pane) gridPaneCuadricula.getChildren().get(fila
* gridPaneCuadricula.getColumnCount() + columna);
        return (CeldaController) panelCelda.getUserData(); // Asumiendo
que has guardado el CeldaController como userData
    }


}
```

3. Clase MenuController

```
package unam.fesaragon.estructuradatos.controladores.vistas;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.SplitPane;
```

```java
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import javafx.scene.text.Text;


public class MenuController {
    public static final String urlMenuController =
"/unam/fesaragon/estructuradatos/vistas/Menu.fxml";

    @FXML
    private AnchorPane aPContenedorMenu;
    @FXML
    private SplitPane splitPane;
    @FXML
    private Text textTitulo1;
    @FXML
    private Text textTitulo2;
    @FXML
    private Button boton1;
    @FXML
    private Button boton2;
    @FXML
    private ImageView imgBoton1;
    @FXML
    private ImageView imgBoton2;

    public SplitPane getSplitPane() {
        return splitPane;
    }

    public AnchorPane getaPContenedorMenu() {
        return aPContenedorMenu;
    }

    public Text getTextTitulo1() {
        return textTitulo1;
    }

    public Text getTextTitulo2() {
        return textTitulo2;
    }

    public Button getBoton1() {
        return boton1;
    }

    public Button getBoton2() {
        return boton2;
    }
```

```
    public ImageView getImgBoton1() {
        return imgBoton1;
    }

    public ImageView getImgBoton2() {
        return imgBoton2;
    }

    public void cambiarImageView(String rutaImagen, ImageView componente)
{
        Image nuevaImagen = new
Image(getClass().getResourceAsStream(rutaImagen));
        componente.setImage(nuevaImagen);
    }
}
```

4. Clase Laberinto

```
package unam.fesaragon.estructuradatos.controladores;

import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.scene.control.SplitPane;
import javafx.stage.Stage;
import javafx.util.Duration;
import
unam.fesaragon.estructuradatos.controladores.vistas.CeldaController;
import unam.fesaragon.estructuradatos.modelos.LaberintoLogica;
import unam.fesaragon.estructuradatos.modelos.adts.ADTArray2D;
import unam.fesaragon.estructuradatos.modelos.excepciones.ArchivoFXML;
import unam.fesaragon.estructuradatos.modelos.javafx.CuadriculaFX;
import unam.fesaragon.estructuradatos.modelos.javafx.Escena;
import unam.fesaragon.estructuradatos.modelos.laberinto.Coordenada;
import unam.fesaragon.estructuradatos.modelos.laberinto.GridLaberinto;
import unam.fesaragon.estructuradatos.vistas.componentes.Vista;

import javax.swing.*;
import java.util.Queue;


public class Laberinto {
    private Vista vista;
    private LaberintoLogica laberintoLogica;
    private SplitPane splitPaneMenuInicio;
    private SplitPane splitPaneParaCargarLaberinto;
    private Coordenada coordenadaDeSalida;
    private Coordenada coordenadaDeEntrada;
    private boolean menuIniciarLaberinto = true;
```

```
    private boolean seEstaPintando = false;
    private boolean yaSePintoUnaVez = false;
    private double seguntosParaPintarCadaCelda = 0.1;

    public Laberinto(int filas, int columnas) throws ArchivoFXML {
        this.vista = new Vista(filas, columnas);
    }

    public Laberinto(int filas, int columnas, double
seguntosParaPintarCadaCelda) throws ArchivoFXML {
        this.vista = new Vista(filas, columnas);
        this.seguntosParaPintarCadaCelda = seguntosParaPintarCadaCelda;
    }

    public void comenzar() {
        Escena escena = new Escena("El juego de la vida", new Stage());

escena.cambiarEscena(vista.getMenuDeInicio().getContenedorMenuController()
);
        configurarBotonesDeLosMenus();

vista.getMenuDeInicio().getMenuController().getSplitPane().getItems().get(
1).setStyle("-fx-background-color: #FFF;");

vista.getMenuParaCargarElLaberinto().getMenuController().getSplitPane().ge
tItems().get(1).setStyle("-fx-background-color: #FFF;");
    }

    private void configurarBotonesDeLosMenus() {
        splitPaneMenuInicio = (SplitPane)
vista.getMenuDeInicio().getMenuController().getSplitPane();
        splitPaneParaCargarLaberinto = (SplitPane)
vista.getMenuParaCargarElLaberinto().getMenuController().getSplitPane();

vista.getMenuDeInicio().getMenuController().getBoton2().setOnAction(event
-> cambiarMenu());

vista.getMenuDeInicio().getMenuController().getBoton1().setOnAction(event
-> iniciarLaberinto());

vista.getMenuParaCargarElLaberinto().getMenuController().getBoton2().setOn
Action(event -> cambiarMenu());

vista.getMenuParaCargarElLaberinto().getMenuController().getBoton1().setOn
Action(event -> {
            try {
                cargarLaberinto();
            } catch (ArchivoFXML e) {
                throw new RuntimeException(e);
            }
```

```
        });
    }

    private void iniciarLaberinto() {
        //Validaciones
        if (seEstaPintando) {
            JOptionPane.showMessageDialog(null, "Se esta pintando el
laberinto", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (coordenadaDeEntrada == null) {
            JOptionPane.showMessageDialog(null, "No haz cargado el
laberinto", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        CuadriculaFX cuadriculaFX =
vista.getMenuDeInicio().getCuadriculaFX();
        int filas = cuadriculaFX.getFilas();
        int columnas = cuadriculaFX.getColumnas();
        GridLaberinto aux = new GridLaberinto(filas, columnas);
        aux.cargarCoordenadaDeEntradaYSalida(coordenadaDeEntrada,
coordenadaDeSalida);
        ADTArray2D<Coordenada> paredes = new
ADTArray2D<>(Coordenada.class, filas, columnas);
        //Obtener el GridLaberinto para pasarlo a la logica.
        for (int fila = 0; fila < filas; fila++) {
            for (int columna = 0; columna < columnas; columna++) {
                paredes.set_item(fila, columna,
cuadriculaFX.getCuadriculaController().getCelda(fila,
columna).getCoordenada());

            }
        }
        aux.cargarParedesDeLaberinto(paredes);
        this.laberintoLogica = new LaberintoLogica(aux);
        if (yaSePintoUnaVez) {
            for (int fila = 0; fila < filas; fila++) {
                for (int columna = 0; columna < columnas; columna++) {
                    if
(cuadriculaFX.getCuadriculaController().getCelda(fila,
columna).getCoordenada().isEstado()) {

cuadriculaFX.getCuadriculaController().getCelda(fila,
columna).getPanelCelda().setStyle("-fx-background-color: #FFF;");
                    }
                }
            }
        }
```

```
        pintarCamino();
    }

    private void pintarCamino() {
        seEstaPintando = true;

vista.getMenuDeInicio().getCuadriculaFX().getCuadriculaController().getCel
da(coordenadaDeEntrada.getFila(),
coordenadaDeEntrada.getColumna()).getPanelCelda().setStyle("-fx-
background-color: #5486b2;");
        Queue<Coordenada> movimientos = laberintoLogica.getMovimientos();
        Timeline timeline = new Timeline(new
KeyFrame(Duration.seconds(seguntosParaPintarCadaCelda), event -> {
            if (!movimientos.isEmpty()) {
                Coordenada coordenadaMovimiento = movimientos.poll();
                CeldaController celda =
vista.getMenuDeInicio().getCuadriculaFX().getCuadriculaController().getCel
da(coordenadaMovimiento.getFila(), coordenadaMovimiento.getColumna());
                if (coordenadaMovimiento.isEstado()) {
                    celda.getPanelCelda().setStyle("-fx-background-color:
#6CB254FF;"); // Verde para avanzar
                    if (movimientos.isEmpty()) {

vista.getMenuDeInicio().getCuadriculaFX().getCuadriculaController().getCel
da(coordenadaDeSalida.getFila(),
coordenadaDeSalida.getColumna()).getPanelCelda().setStyle("-fx-background-
color: #5486b2;");
                        seEstaPintando = false;
                        yaSePintoUnaVez = true;
                    }
                }
                // Si el movimiento es retroceder (estado es false), se
pinta de rojo
                else {
                    celda.getPanelCelda().setStyle("-fx-background-color:
#b25454;"); // Rojo para retroceder
                }
            }
        }));
        timeline.setCycleCount(Timeline.INDEFINITE);
        timeline.play();
    }

    private void cargarLaberinto() throws ArchivoFXML {
        CuadriculaFX copia =
clonarCuadricula(vista.getMenuParaCargarElLaberinto().getCuadriculaFX());
        vista.getMenuDeInicio().setCuadriculaFX(copia);
        // Actualizar visualmente el contenedor

vista.getMenuDeInicio().getStackPaneDeCuadriculaFX().getChildren().clear()
```

```
;

vista.getMenuDeInicio().getStackPaneDeCuadriculaFX().getChildren().add(cop
ia.getCuadriculaController().getGridPaneCuadricula());

vista.getMenuDeInicio().getContenedorMenuController().requestLayout();
        coordenadaDeEntrada = cuadroParaIngresarCoordenada("La coordenada
de entrada:", "Coordenada de entrada", JOptionPane.QUESTION_MESSAGE);
        coordenadaDeSalida = cuadroParaIngresarCoordenada("La coordenada
de salida:", "Coordenada de salida", JOptionPane.QUESTION_MESSAGE);
        cambiarMenu();
    }

    public Coordenada cuadroParaIngresarCoordenada(String mensaje, String
titulo, int tipoDeMensaje) {
        boolean datoValido = false;
        Coordenada coordenada = null;

        while (!datoValido) {
            String ingresado = JOptionPane.showInputDialog(null, mensaje,
titulo, tipoDeMensaje);
            if (ingresado == null || ingresado.isEmpty()) {
                JOptionPane.showMessageDialog(null, "Coordenada no válida.
Por favor, inténtalo de nuevo.", "Error", JOptionPane.ERROR_MESSAGE);
            } else if (ingresado.matches("\\d+,\\d+")) {
                String[] partes = ingresado.split(",");
                try {
                    int filaInsertada =
Integer.parseInt(partes[0].trim());
                    int columnaInsertada =
Integer.parseInt(partes[1].trim());
                    if ((filaInsertada >= 0 && filaInsertada <
vista.getMenuDeInicio().getCuadriculaFX().getFilas()) && (columnaInsertada
>= 0 && columnaInsertada <
vista.getMenuDeInicio().getCuadriculaFX().getColumnas())) {
                        coordenada = new Coordenada(filaInsertada,
columnaInsertada);
                        datoValido = true;
                    } else {
                        JOptionPane.showMessageDialog(null, "Las
coordenadas que ingresaste estan fuera de rango", "Error",
JOptionPane.ERROR_MESSAGE);
                    }
                } catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "Formato
incorrecto. Por favor, ingresa en el formato '3,2'", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(null, "Formato incorrecto.
```

```
Por favor, ingresa en el formato '3,2'", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
        return coordenada;
    }


    private void cambiarMenu() {

vista.getMenuDeInicio().getContenedorMenuController().getChildren().clear(
);
        if (menuIniciarLaberinto) {
            if (seEstaPintando) {
                JOptionPane.showMessageDialog(null, "Se esta pintando el
laberinto", "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }

vista.getMenuDeInicio().getContenedorMenuController().getChildren().add(sp
litPaneParaCargarLaberinto);
            menuIniciarLaberinto = false;
        } else {

vista.getMenuDeInicio().getContenedorMenuController().getChildren().add(sp
litPaneMenuInicio);
            menuIniciarLaberinto = true;
        }
    }

    //Copia la grid del menu para cargar el laberinto para mostrarlo al
menu de inicio
    private CuadriculaFX clonarCuadricula(CuadriculaFX original) throws
ArchivoFXML {
        CuadriculaFX copia = new CuadriculaFX(original.getFilas(),
original.getColumnas());
        for (int fila = 0; fila < original.getFilas(); fila++) {
            for (int columna = 0; columna < original.getColumnas();
columna++) {
                CeldaController celdaOriginal =
original.getCuadriculaController().getCelda(fila, columna);
                Coordenada coordenadaOriginal =
celdaOriginal.getCoordenada();
                CeldaController celdaCopia =
copia.getCuadriculaController().getCelda(fila, columna);
                Coordenada coordenadaCopia = new
Coordenada(coordenadaOriginal.getFila(), coordenadaOriginal.getColumna(),
coordenadaOriginal.isEstado());
                celdaCopia.setCoordenada(coordenadaCopia);
```

```
celdaCopia.getPanelCelda().setStyle(celdaOriginal.getPanelCelda().getStyle
());
                }
            }
            return copia;
        }


        public LaberintoLogica getLaberintoLogica() {
            return laberintoLogica;
        }

}
```

5. Clase ADTArray2D

```
package unam.fesaragon.estructuradatos.modelos.adts;

import java.lang.reflect.Array;

public class ADTArray2D<T> {
    private int filas;
    private int columnas;
    private T array[][];

    //  Constructor de la clase
    public ADTArray2D(Class<T> clazz, int filas, int columnas) {
        this.filas = filas;
        this.columnas = columnas;
        this.array = (T[][]) Array.newInstance(clazz, filas, columnas);
    }
    //  Metodo limpiar
    public void clear(T dato) {
        for (int i = 0; i < filas; i++) {
            for (int j = 0; j < columnas; j++) {
                this.array[i][j] = dato;
            }
        }
    }

    //  Colocar item en la coordenada específica
    public void set_item(int renglon, int columna, T dato){
        this.array[renglon][columna] = dato;
    }

    //   Obtener item en la coordenada específica
    public T get_item(int renglon, int columna){
        return this.array[renglon][columna];
    }
```

```
    //Copiar estado de otro array
    public void copiarEstadoDe(ADTArray2D<T> arrayACopiar){
        for (int filas = 0; filas < arrayACopiar.getFilas(); filas++) {
            for (int columnas = 0; columnas < arrayACopiar.getColumnas();
columnas++) {
                this.array[filas][columnas] =
arrayACopiar.get_item(filas,columnas);
            }
        }
    }
    public int getFilas() {
        return filas;
    }

    public int getColumnas() {
        return columnas;
    }

    @Override
    public String toString() {
        String matriz= "";
        for (T[] elemento : this.array) {
            for (T el : elemento) {
                matriz= matriz+el+" ";
            }
            matriz+="\n";
        }
        return matriz;
    }
}
```

6. Clase ADTStack

```
package unam.fesaragon.estructuradatos.modelos.adts;

public class ADTStack<T> {
    private ListaDoblementeLigada<T> datos;

    public ADTStack() {
        datos = new ListaDoblementeLigada<T>();
    }
    public boolean isEmpty(){
        return datos.esta_vacia();
    }
    public int length(){
        return datos.get_tamanio();
    }
    public T pop(){
```

```
        T datoASacar = datos.obtener(datos.get_tamanio()-1);
        datos.eliminar_el_final();
        return datoASacar;
    }
    public T peek(){
        return datos.obtener(datos.get_tamanio()-1);
    }
    public void push(T datoAInsertar){
        datos.agregar_al_final(datoAInsertar);
    }

    public void imprimirStack(){
        datos.transversal();
    }
}
```

7. Clase ColaADT

```
package unam.fesaragon.estructuradatos.modelos.adts;

public class ColaADT<T> {
    private ListaDoblementeLigada<T> data;

    public ColaADT() {
        this.data = new ListaDoblementeLigada<>();
    }

    public boolean estaVacia() {
        return this.data.esta_vacia();
    }

    public int longitud() {
        return this.data.get_tamanio();
    }

    public T frente() {
        return this.data.obtener(0);
    }

    public void encolar(T valor) { //enqueue
        this.data.agregar_al_final(valor);
    }

    public T desEncolar() {
        if (this.data.esta_vacia()) {
            System.out.println("La cola está vacía");
        }
        T dato = this.data.obtener(0);
        this.data.eliminar_el_primero();
```

```
        return dato;
    }


    public T siguiente() {
        if (this.data.get_tamanio() < 2) {
            System.out.println("No hay un segundo elemento en la cola");
        }
        return this.data.obtener(1);
    }


    @Override
    public String toString() {
        return this.data.toString();
    }
}
```

8. Clase ListaDoblementeLigada

```
package unam.fesaragon.estructuradatos.modelos.adts;

public class ListaDoblementeLigada<T> {
    private NodoDoble<T> head;
    private NodoDoble<T> tail;
    private int tamanio;

    // Constructor
    public ListaDoblementeLigada() {
        this.head = null;
        this.tail = null;
        this.tamanio = 0;
    }

    public ListaDoblementeLigada(NodoDoble<T> head, NodoDoble<T> tail) {
        this.head = head;
        this.tail = tail;
    }

    // Comprobar si está vacía
    public boolean esta_vacia() {
        return this.tamanio == 0;
    }

    // Agregar al inicio de la lista
    public void agregar_al_inicio(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (esta_vacia()) {
            this.head = nuevo;
```

```
            this.tail = nuevo;
        } else {
            nuevo.setSiguiente(this.head);
            this.head.setAnterior(nuevo);
            this.head = nuevo;
        }
        tamanio++;
    }

    // Agregar al final de la lista
    public void agregar_al_final(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (esta_vacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            this.tail.setSiguiente(nuevo);
            nuevo.setAnterior(this.tail);
            this.tail = nuevo;
        }
        tamanio++;
    }

    // Agregar después de un nodo de referencia
    public void agregar_después_de(T referencia, T valor) {
        if (esta_vacia()) {
            System.out.println("La lista esta vacia y por lo tanto no
existe la referencia");
            return;
        }

        NodoDoble<T> aux = this.head;
        while (aux != null) {
            if (aux.getDato().equals(referencia)) {
                break;
            }
            aux = aux.getSiguiente();
        }

        if (aux == null) {
            System.out.println("El nodo de referencia no existe");
            return;
        }

        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        nuevo.setSiguiente(aux.getSiguiente());
        nuevo.setAnterior(aux);
        //Nodo de referencia, no es la ultima en la lista.
        if (aux.getSiguiente() != null) {
            aux.getSiguiente().setAnterior(nuevo);
```

```
        } else {
            this.tail = nuevo;
        }
        aux.setSiguiente(nuevo);
        this.tamanio++;
    }

    // Obtener el elemento en una posición específica
    public T obtener(int posicion) {
        //Condiciones
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return null;
        }
        if (posicion > this.tamanio || posicion<0) {
            System.out.println("Posición fuera de rango");
            return null;
        }

        NodoDoble<T> aux = this.head;
        for (int i = 0; i < posicion; i++) {
            aux = aux.getSiguiente();
        }

        return aux.getDato();
    }

    // Eliminar el primer elemento
    public void eliminar_el_primero() {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return;
        }
        if (this.head == this.tail) { // Solo un elemento
            this.head = null;
            this.tail = null;
        } else {
            this.head = this.head.getSiguiente();
            this.head.setAnterior(null);
        }
        tamanio--;
    }

    // Eliminar el último elemento
    public void eliminar_el_final() {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return;
        }
```

```
        if (this.head == this.tail) { // Solo un elemento
            this.head = null;
            this.tail = null;
        } else {
            this.tail = this.tail.getAnterior();
            this.tail.setSiguiente(null);
        }
        tamanio--;
    }

    // Eliminar un elemento en una posición específica
    public void eliminar(int posicion) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }
        if (posicion >= this.tamanio || posicion < 0) {
            System.out.println("Posición fuera de rango");
            return;
        }

        if (posicion == 0) {
            eliminar_el_primero();
            return;
        }
        if (posicion == this.tamanio - 1) {
            eliminar_el_final();
            return;
        }

        NodoDoble<T> aux = this.head;
        int i = 0;
        while (i < posicion) {
            aux = aux.getSiguiente();
            i++;
        }

        aux.getAnterior().setSiguiente(aux.getSiguiente());
        aux.getSiguiente().setAnterior(aux.getAnterior());
        this.tamanio--;
    }


    // Buscar un elemento y retornar su posición
    public int buscar(T valor) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return 0;
        }
```

```
        NodoDoble<T> aux = this.head;
        int posicion = 0;
        while (aux != null) {
            if (aux.getDato().equals(valor)) {
                return posicion;
            }
            aux = aux.getSiguiente();
            posicion++;
        }
        System.out.println("No se encontro el elemento");
        return 0;
    }

    // Actualizar un elemento
    public void actualizar(T a_buscar, T valor) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }

        NodoDoble<T> aux = this.head;
        while (aux != null) {
            if (aux.getDato().equals(a_buscar)) {
                aux.setDato(valor);
                return;
            }
            aux = aux.getSiguiente();
        }
        System.out.println("El elemento valor o elemento no esta en la
lista");
    }

    // Recorrido transversal en una dirección específica
    // Verdadero es valor por defecto y con el falso es de derecha a
izquierda
    public void transversal(boolean porDefecto) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }

        if (porDefecto) {
            NodoDoble<T> aux = this.head;
            while (aux != null) {
                System.out.print("[" + aux.getDato() + "] <--> ");
                aux = aux.getSiguiente();
            }
        } else {
            NodoDoble<T> aux = this.tail;
            while (aux != null) {
```

```
                System.out.print("[" + aux.getDato() + "] <--> ");
                aux = aux.getAnterior();
            }
        }
        System.out.println("NULL");
    }
    //Sobrecargue el metodo para poner por default el recorrido de
izquierda a derecha
    public void transversal() {
        transversal(true);
    }

    //GETTERS Y SETTERS
    // Obtener el tamaño de la lista
    public int get_tamanio() {
        return this.tamanio;
    }

    //ToString
    @Override
    public String toString() {
        StringBuilder estado = new StringBuilder();
        NodoDoble<T> aux = this.head;
        while (aux != null) {
            estado.append("[").append(aux.getDato()).append("] <--> ");
            aux = aux.getSiguiente();
        }
        return estado +" Tamaño: "+ this.tamanio;
    }
}
```

9. Clase NodoDoble

```
package unam.fesaragon.estructuradatos.modelos.adts;

public class NodoDoble<T> {
    private T dato;
    private NodoDoble<T> anterior;
    private NodoDoble<T> siguiente;

    // Constructores
    public NodoDoble() {
    }

    public NodoDoble(T dato) {
        this.dato = dato;
        this.anterior = null;
        this.siguiente = null;
    }
```

```
    public NodoDoble(T dato, NodoDoble<T> anterior, NodoDoble<T>
siguiente) {
        this.dato = dato;
        this.anterior = anterior;
        this.siguiente = siguiente;
    }

    // Getters y Setters
    public T getDato() {
        return dato;
    }

    public void setDato(T dato) {
        this.dato = dato;
    }

    public NodoDoble<T> getAnterior() {
        return anterior;
    }

    public void setAnterior(NodoDoble<T> anterior) {
        this.anterior = anterior;
    }

    public NodoDoble<T> getSiguiente() {
        return siguiente;
    }

    public void setSiguiente(NodoDoble<T> siguiente) {
        this.siguiente = siguiente;
    }
}
```

10. Clase ArchivoFXML (Exception)

```
package unam.fesaragon.estructuradatos.modelos.excepciones;

import java.io.IOException;

public class ArchivoFXML extends IOException {
    public ArchivoFXML(IOException message) {
        super(message);
    }
}
```

11. Clase CuadriculaFX

```java
package unam.fesaragon.estructuradatos.modelos.javafx;

import javafx.fxml.FXMLLoader;
import javafx.scene.layout.*;
import
unam.fesaragon.estructuradatos.controladores.vistas.CeldaController;
import
unam.fesaragon.estructuradatos.controladores.vistas.CuadriculaController;
import unam.fesaragon.estructuradatos.modelos.adts.ColaADT;
import unam.fesaragon.estructuradatos.modelos.excepciones.ArchivoFXML;
import unam.fesaragon.estructuradatos.modelos.laberinto.Coordenada;

import java.io.IOException;

public class CuadriculaFX {
    //Componentes de la cuadricula
    private CuadriculaController cuadriculaController;
    private boolean habilitarTouchDeceldas = false;

    //Atributos de la cuadricula
    private int filas;
    private int columnas;

    public CuadriculaFX(int filas, int columnas) throws ArchivoFXML {
        this.filas = filas;
        this.columnas = columnas;
        cargarComponentes();
    }

    public CuadriculaFX(int filas, int columnas, boolean
habilitarTouchDeCeldas) throws ArchivoFXML {
        this.filas = filas;
        this.columnas = columnas;
        this.habilitarTouchDeceldas = habilitarTouchDeCeldas;
        cargarComponentes();
    }

    private void cargarComponentes() throws ArchivoFXML {
        FXMLLoader loaderCuadricula = new
FXMLLoader(getClass().getResource(CuadriculaController.urlFXMLDeCuadricula
));
        //Cargando la cuadricula
        try {
            loaderCuadricula.load();
        } catch (IOException e) {
            throw new ArchivoFXML(e);
        }
        cuadriculaController = loaderCuadricula.getController();
        //Configurar el numero de celdas de la cuadriculaController.
        configurarCuadriculaConElTamano();
```

```
        llenarCuadriculaDeCeldas();
        cuadriculaController.getGridPaneCuadricula().setMaxSize(600.0,
600.0);

    }

    private void configurarCuadriculaConElTamano() {

cuadriculaController.getGridPaneCuadricula().getRowConstraints().clear();

cuadriculaController.getGridPaneCuadricula().getColumnConstraints().clear(
);
        double rowHeight = 30.0; // Altura de cada fila
        double columnWidth = 30.0; // Ancho de cada columna

        for (int fila = 0; fila < this.getFilas(); fila++) {
            RowConstraints rowConstraints = new RowConstraints();
            rowConstraints.setPrefHeight(rowHeight);
            rowConstraints.setVgrow(Priority.ALWAYS);

cuadriculaController.getGridPaneCuadricula().getRowConstraints().add(rowCo
nstraints);
        }

        for (int columna = 0; columna < this.getColumnas(); columna++) {
            ColumnConstraints columnConstraints = new ColumnConstraints();
            columnConstraints.setPrefWidth(columnWidth);
            columnConstraints.setHgrow(Priority.ALWAYS);

cuadriculaController.getGridPaneCuadricula().getColumnConstraints().add(co
lumnConstraints);
        }

        // Permitir el crecimiento dinámico del GridPane

cuadriculaController.getGridPaneCuadricula().setMaxSize(Double.MAX_VALUE,
Double.MAX_VALUE);

cuadriculaController.getGridPaneCuadricula().setPrefSize(columnWidth *
this.getColumnas(), rowHeight * this.getFilas());
    }

    private void llenarCuadriculaDeCeldas() throws ArchivoFXML {
        ColaADT<CeldaController> celdas = colaDeCeldas();
        for (int fila = 0; fila < this.getFilas(); fila++) {
            for (int columna = 0; columna < this.getColumnas(); columna++)
{
                CeldaController celdaController = celdas.desEncolar();
                Pane panelCelda = celdaController.getPanelCelda();
                panelCelda.setMaxSize(Double.MAX_VALUE, Double.MAX_VALUE);
```

```
                GridPane.setHgrow(panelCelda, Priority.ALWAYS);
                GridPane.setVgrow(panelCelda, Priority.ALWAYS);
                celdaController.setCoordenada(new Coordenada(fila,
columna));
                panelCelda.setUserData(celdaController); // Guardar el
CeldaController como userData

cuadriculaController.getGridPaneCuadricula().add(panelCelda, columna,
fila);
            }
        }
    }


    private ColaADT<CeldaController> colaDeCeldas() throws ArchivoFXML {
        ColaADT<CeldaController> celdas = new ColaADT<>();
        //Cargando la celda
        for (int cantidadDeCeldas = 0; cantidadDeCeldas < (filas *
columnas); cantidadDeCeldas++) {
            FXMLLoader loaderCelda = new
FXMLLoader(getClass().getResource(CeldaController.urlFXMLDeCelda));
            try {
                loaderCelda.load();
            } catch (IOException e) {
                throw new ArchivoFXML(e);
            }
            CeldaController celdaController = loaderCelda.getController();
            if (!habilitarTouchDeceldas)
celdaController.setCambiarEstadoDeLasCeldas_Click(false);
            celdas.encolar(celdaController);
        }
        return celdas;
    }

    //GETTERS Y SETTERS
    public int getFilas() {
        return filas;
    }

    public int getColumnas() {
        return columnas;
    }

    public CuadriculaController getCuadriculaController() {
        return cuadriculaController;
    }

    public boolean isHabilitarTouchDeceldas() {
        return habilitarTouchDeceldas;
```

```
    }
}
```

12. Clase Escena

```
package unam.fesaragon.estructuradatos.modelos.javafx;

import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

public class Escena {
    private String titulo;
    private String urlImageFXMLTitle;
    private Scene escenaCargada;
    private Stage stagePrincipal;

    public Escena(Stage stagePrincipal) {
        this.stagePrincipal = stagePrincipal;
    }

    public Escena(String titulo, Stage stagePrincipal) {
        this.titulo = titulo;
        this.stagePrincipal = stagePrincipal;
        this.stagePrincipal.setTitle(titulo);
    }

    public void cambiarEscena(AnchorPane componenteAMostrar) {
        try {
            escenaCargada = new Scene(componenteAMostrar);
            stagePrincipal.setTitle(titulo);
            stagePrincipal.setScene(escenaCargada);
            stagePrincipal.show();
        } catch (Exception e) {
            System.out.println(e);
        }

    }


    public Scene getEscenaCargada() {
        return escenaCargada;
    }


    public String getTitulo() {
        return titulo;
    }
```

```
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getUrlImageFXMLTitle() {
        return urlImageFXMLTitle;
    }

    public void setUrlImageFXMLTitle(String urlImageFXMLTitle) {
        this.urlImageFXMLTitle = urlImageFXMLTitle;
    }
}
```

13. Clase MenuFX

```
package unam.fesaragon.estructuradatos.modelos.javafx;

import javafx.fxml.FXMLLoader;
import javafx.geometry.Pos;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.StackPane;
import unam.fesaragon.estructuradatos.controladores.vistas.MenuController;
import unam.fesaragon.estructuradatos.modelos.excepciones.ArchivoFXML;

import java.io.IOException;

public class MenuFX {
    //Elementos del menuFX
    private AnchorPane contenedorMenuController;
    private MenuController menuController;
    //Elementos de la cuadriculaFX
    private CuadriculaFX cuadriculaFX;
    //Elementos de menuParaCargarCuadricula
    StackPane stackPaneDeCuadriculaFX;


    public MenuFX(int filas, int columnas) throws ArchivoFXML {
        this.cuadriculaFX = new CuadriculaFX(filas, columnas);
        cargarComponentes();
        ingresarCuadriculaFXAlMenu();
    }

    /**
     * @param esMenuParaCargarElLaberinto Si el valor es true, se
configura para ser un menu para cargar el laberinto
     */
    public MenuFX(int filas, int columnas, boolean
esMenuParaCargarElLaberinto) throws ArchivoFXML {
```

```
        this.cuadriculaFX = new CuadriculaFX(filas, columnas,
esMenuParaCargarElLaberinto);
        cargarComponentes();
        ingresarCuadriculaFXAlMenu();
        configurarMenuFXParaCargarLaberinto();
    }

    private void configurarMenuFXParaCargarLaberinto() {
        getMenuController().getTextTitulo1().setText("Cargar Laberinto");
        getMenuController().getTextTitulo2().setText("Regresar al Menu");
    }

    private void ingresarCuadriculaFXAlMenu() {
        stackPaneDeCuadriculaFX = new StackPane();
        stackPaneDeCuadriculaFX.setStyle("-fx-background-color:
#93C7C1;");
        stackPaneDeCuadriculaFX.setPrefSize(Double.MAX_VALUE,
Double.MAX_VALUE);
        // Añádiendo el GridPane al StackPane

stackPaneDeCuadriculaFX.getChildren().add(cuadriculaFX.getCuadriculaContro
ller().getGridPaneCuadricula());
        // Centrando el GridPane dentro del StackPane

StackPane.setAlignment(cuadriculaFX.getCuadriculaController().getGridPaneC
uadricula(), Pos.CENTER);
        //Envolver el AnchorPane al StackPane
        AnchorPane.setTopAnchor(stackPaneDeCuadriculaFX, 0.0);
        AnchorPane.setBottomAnchor(stackPaneDeCuadriculaFX, 0.0);
        AnchorPane.setLeftAnchor(stackPaneDeCuadriculaFX, 0.0);
        AnchorPane.setRightAnchor(stackPaneDeCuadriculaFX, 0.0);
        //Remplazar el StackPane
        this.getMenuController().getSplitPane().getItems().set(0,
stackPaneDeCuadriculaFX);
    }


    private void cargarComponentes() throws ArchivoFXML {
        FXMLLoader loaderCuadricula = new
FXMLLoader(getClass().getResource(MenuController.urlMenuController));
        try {
            contenedorMenuController = loaderCuadricula.load();
        } catch (IOException e) {
            throw new ArchivoFXML(e);
        }
        menuController = loaderCuadricula.getController();
    }


    public MenuController getMenuController() {
```

```
        return menuController;
    }

    public AnchorPane getContenedorMenuController() {
        return contenedorMenuController;
    }

    public CuadriculaFX getCuadriculaFX() {
        return cuadriculaFX;
    }

    public void setCuadriculaFX(CuadriculaFX cuadriculaFX) {
        this.cuadriculaFX = cuadriculaFX;
    }

    public StackPane getStackPaneDeCuadriculaFX() {
        return stackPaneDeCuadriculaFX;
    }
}
```

14. Clase Coordenada

```
package unam.fesaragon.estructuradatos.modelos.laberinto;

import java.util.Objects;

public class Coordenada {
    private int fila;
    private int columna;
    private boolean estado;

    public Coordenada(int fila, int columna, boolean estado) {
        this.fila = fila;
        this.columna = columna;
        this.estado = estado;
    }

    public Coordenada(int fila, int columna) {
        this.fila = fila;
        this.columna = columna;
        this.estado = true;
    }

    public Coordenada() {
        this.estado = true;
    }

    public int getFila() {
        return fila;
```

```
    }

    public void setFila(int fila) {
        this.fila = fila;
    }

    public int getColumna() {
        return columna;
    }

    public void setColumna(int columna) {
        this.columna = columna;
    }

    public boolean isEstado() {
        return estado;
    }

    public void setEstado(boolean estado) {
        this.estado = estado;
    }

    @Override
    public String toString() {
        return "Coordenada[" +
                "fila=" + fila +
                ", columna=" + columna +
                ", estado=" + estado +
                ']';
    }

    @Override

    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Coordenada that = (Coordenada) o;
        return fila == that.fila && columna == that.columna;
    }

    @Override
    public int hashCode() {
        return Objects.hash(fila, columna);
    }
}
```

15. Clase GridLaberinto

```java
package unam.fesaragon.estructuradatos.modelos.laberinto;

import unam.fesaragon.estructuradatos.modelos.adts.ADTArray2D;

public class GridLaberinto {
    private ADTArray2D<Coordenada> grid;
    private Coordenada esquinaSuperiorIzquierda;
    private Coordenada esquinaInferiorDerecha;
    private Coordenada coordenadaDeEntrada;
    private Coordenada coordenadaDeSalida;

    public GridLaberinto(int filas, int columnas) {
        this.grid = new ADTArray2D<>(Coordenada.class, filas, columnas);
        iniciarCoordenadas();
    }

    private void iniciarCoordenadas() {
        for (int filas = 0; filas < grid.getFilas(); filas++) {
            for (int columnas = 0; columnas < grid.getColumnas();
columnas++) {
                this.grid.set_item(filas, columnas, new Coordenada(filas,
columnas));
            }
        }
        esquinaSuperiorIzquierda = new Coordenada(0, 0);
        esquinaInferiorDerecha = new Coordenada(grid.getFilas() - 1,
grid.getColumnas() - 1);
    }

    public void cargarParedesDeLaberinto(ADTArray2D<Coordenada> paredes) {
        if (this.grid.getFilas() != paredes.getFilas() &&
this.grid.getColumnas() != paredes.getFilas()) {
            System.out.println("Las columnas o filas no coinciden con la
grid");
            return;
        }
        for (int filas = 0; filas < this.grid.getFilas(); filas++) {
            for (int columnas = 0; columnas < this.grid.getColumnas();
columnas++) {
                //Comparacion de estados en ambas grid. Grid verdadero y
paredesGrid falso
                if (grid.get_item(filas, columnas).isEstado() &&
!paredes.get_item(filas, columnas).isEstado()) {
                    grid.get_item(filas, columnas).setEstado(false);
                }
            }
        }

    }
```

```
    public void cargarCoordenadaDeEntradaYSalida(Coordenada
coordenadaDeEntrada, Coordenada coordenadaDeSalida) {
        this.coordenadaDeEntrada = coordenadaDeEntrada;
        this.coordenadaDeSalida = coordenadaDeSalida;
    }

    public Coordenada getCoordenada(int fila, int columna) {
        return grid.get_item(fila, columna);
    }

    public Coordenada getEsquinaSuperiorIzquierda() {
        return esquinaSuperiorIzquierda;
    }

    public Coordenada getEsquinaInferiorDerecha() {
        return esquinaInferiorDerecha;
    }

    public Coordenada getCoordenadaDeEntrada() {
        return coordenadaDeEntrada;
    }

    public Coordenada getCoordenadaDeSalida() {
        return coordenadaDeSalida;
    }

    @Override
    public String toString() {
        return "GridLaberinto{" +
                "grid=" + grid +
                '}';
    }
}
```

16. Clase LaberintoLogica

```
package unam.fesaragon.estructuradatos.modelos;

import unam.fesaragon.estructuradatos.modelos.adts.ADTStack;
import unam.fesaragon.estructuradatos.modelos.adts.ColaADT;
import unam.fesaragon.estructuradatos.modelos.laberinto.Coordenada;
import unam.fesaragon.estructuradatos.modelos.laberinto.GridLaberinto;

import java.util.HashSet;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Set;

public class LaberintoLogica {
```

```
    private GridLaberinto gridLaberintoConParedes;
    private ADTStack<Coordenada> camino;
    private Set<Coordenada> visitadas;
    private Queue<Coordenada> movimientos;

    //gridLaberintoConParedes debe estar previamente configurada:
    //Coordenada de entrada y salida
    //Paredes ya establecidas en false
    //Camino en true, o celdas validas
    public LaberintoLogica(GridLaberinto gridLaberintoConParedes) {
        this.gridLaberintoConParedes = gridLaberintoConParedes;
        this.camino = new ADTStack<>();
        this.visitadas = new HashSet<>();
        this.movimientos = new LinkedList<>();
        Coordenada entrada =
this.gridLaberintoConParedes.getCoordenadaDeEntrada();
        this.camino.push(entrada);
        this.visitadas.add(entrada);
        obtenerPilaCoordenadasParaLaSalida();
    }

    public void obtenerPilaCoordenadasParaLaSalida() {
        //Cuando el peek de la pila camino sea igual a la coordenada de
salida se detendra
        while
(!camino.peek().equals(gridLaberintoConParedes.getCoordenadaDeSalida())) {
            moverseASiguienteCoordenada(this.camino.peek());
        }
    }

    // Avanzar a la siguiente coordenada
    private void moverseASiguienteCoordenada(Coordenada
coordenadaACalcularLaSiguiente) {
        Coordenada coordenadaSiguiente =
siguienteCoordenada(coordenadaACalcularLaSiguiente);
        if (coordenadaSiguiente == null) {
            // Si no hay salida, marco el retroceso como false
            camino.peek().setEstado(false);
            movimientos.add(camino.peek());
            camino.pop();
        } else {
            coordenadaSiguiente.setEstado(true);
            this.camino.push(coordenadaSiguiente);
            this.visitadas.add(coordenadaSiguiente);
            this.movimientos.add(coordenadaSiguiente);
        }
    }

    private Coordenada siguienteCoordenada(Coordenada coordenadaDeEntrada)
{
```

```
        ColaADT<Coordenada> coordenadasMovimiento = new ColaADT<>();
        // Agregando las posibles coordenadas vecinas en el orden
izquierda, arriba, derecha, abajo
        coordenadasMovimiento.encolar(new
Coordenada(coordenadaDeEntrada.getFila(), coordenadaDeEntrada.getColumna()
- 1));  // Izquierda
        coordenadasMovimiento.encolar(new
Coordenada(coordenadaDeEntrada.getFila() - 1,
coordenadaDeEntrada.getColumna()));  // Arriba
        coordenadasMovimiento.encolar(new
Coordenada(coordenadaDeEntrada.getFila(), coordenadaDeEntrada.getColumna()
+ 1));  // Derecha
        coordenadasMovimiento.encolar(new
Coordenada(coordenadaDeEntrada.getFila() + 1,
coordenadaDeEntrada.getColumna()));  // Abajo

        // Buscando una coordenada válida
        while (!coordenadasMovimiento.estaVacia()) {
            Coordenada candidata = coordenadasMovimiento.frente();
            boolean coordenadaDentroDeLosLimites =
coordenadaDentroDeLosLimites(candidata);

            // Verificando que la coordenada no haya sido visitada antes y
sea válida
            if (coordenadaDentroDeLosLimites) {
                boolean yaHaRecorridoEsaCelda =
visitadas.contains(gridLaberintoConParedes.getCoordenada(candidata.getFila
(), candidata.getColumna()));
                boolean noEsUnaParedEsaCoordenada =
gridLaberintoConParedes.getCoordenada(candidata.getFila(),
candidata.getColumna()).isEstado();
                if (!yaHaRecorridoEsaCelda && noEsUnaParedEsaCoordenada) {
                    return candidata;
                }

            }
            coordenadasMovimiento.desEncolar();  // Descartando la
coordenada no válida
        }
        return null;  // No se encontro una coordenada valida
    }

    private boolean coordenadaDentroDeLosLimites(Coordenada
coordenadaAEvaluar) {
        boolean dentroDeLimitesDeColumnas =
coordenadaAEvaluar.getColumna() >=
gridLaberintoConParedes.getEsquinaSuperiorIzquierda().getColumna() &&
coordenadaAEvaluar.getColumna() <=
gridLaberintoConParedes.getEsquinaInferiorDerecha().getColumna();
        boolean dentroDeLimitesFilas = coordenadaAEvaluar.getFila() >=
```

```
gridLaberintoConParedes.getEsquinaSuperiorIzquierda().getFila() &&
coordenadaAEvaluar.getFila() <=
gridLaberintoConParedes.getEsquinaInferiorDerecha().getFila();
        return dentroDeLimitesDeColumnas && dentroDeLimitesFilas;
    }

    public ADTStack<Coordenada> getCamino() {
        return camino;
    }

    public Queue<Coordenada> getMovimientos() {
        return movimientos;
    }

}
```

17. Clase Vista

```
package unam.fesaragon.estructuradatos.vistas.componentes;

import javafx.scene.image.ImageView;
import unam.fesaragon.estructuradatos.modelos.excepciones.ArchivoFXML;

import unam.fesaragon.estructuradatos.modelos.javafx.MenuFX;

public class Vista {
    MenuFX menuDeInicio;
    MenuFX menuParaCargarElLaberinto;

    public Vista(int filas, int columnas) throws ArchivoFXML {
        this.menuDeInicio = new MenuFX(filas, columnas);
        this.menuParaCargarElLaberinto = new MenuFX(filas, columnas,true);

menuParaCargarElLaberinto.getMenuController().cambiarImageView("/unam/fesa
ragon/estructuradatos/assets/icons/upload2.png",menuParaCargarElLaberinto.
getMenuController().getImgBoton1());

menuParaCargarElLaberinto.getMenuController().cambiarImageView("/unam/fesa
ragon/estructuradatos/assets/icons/back.png",menuParaCargarElLaberinto.get
MenuController().getImgBoton2());

menuParaCargarElLaberinto.getMenuController().getImgBoton2().setRotate(0);

    }

    public MenuFX getMenuDeInicio() {
        return menuDeInicio;
    }
```

```
    public MenuFX getMenuParaCargarElLaberinto() {
        return menuParaCargarElLaberinto;
    }
}
```

18. FXML Celda

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.Pane?>


<Pane fx:id="panelCelda" onMouseClicked="#onClicked" style="-fx-
background-color: #FFF;" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="unam.fesaragon.estructuradatos.controladores.vistas.CeldaCo
ntroller" />
```

19. FXML Cuadricula

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.String?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>

<AnchorPane fx:id="anchorPaneContenedorGridPane"
maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308"
            minHeight="400.0" minWidth="400.0" style="-fx-background-
color: black;"
            xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1"

fx:controller="unam.fesaragon.estructuradatos.controladores.vistas.Cuadric
ulaController">
    <children>
        <GridPane fx:id="gridPaneCuadricula" minHeight="400.0"
minWidth="400.0" AnchorPane.topAnchor="0.0" AnchorPane.bottomAnchor="0.0"
                  AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
stylesheets="@styles.css">
            <columnConstraints>
                <ColumnConstraints minWidth="10.0" maxWidth="50.0"
hgrow="ALWAYS"/>
            </columnConstraints>
            <rowConstraints>
                <RowConstraints minHeight="10.0" maxHeight="50.0"
vgrow="ALWAYS"/>
```

```
            </rowConstraints>
            <styleClass>
                <String fx:value="grid-pane"/>
                <String fx:value="grid-line"/>
            </styleClass>
        </GridPane>
    </children>
</AnchorPane>
```

20. FXML Menu

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.SplitPane?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.StackPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane fx:id="aPContenedorMenu" prefHeight="1080.0"
prefWidth="1920.0" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="unam.fesaragon.estructuradatos.controladores.vistas.MenuCon
troller">
   <children>
      <SplitPane fx:id="splitPane" dividerPositions="0.8248175182481752"
prefHeight="1080.0" prefWidth="1920.0">
        <items>
            <StackPane />
          <AnchorPane maxWidth="-Infinity" minWidth="-Infinity"
prefWidth="240.0" SplitPane.resizableWithParent="false">
              <children>
                 <VBox alignment="CENTER" maxWidth="240.0"
prefWidth="240.0" spacing="20.0" translateY="100.0">
                    <children>
                       <Text fx:id="textTitulo1" strokeType="OUTSIDE"
strokeWidth="0.0" text="Iniciar Laberinto" textAlignment="CENTER"
textOrigin="CENTER">
                          <font>
                             <Font name="Verdana Bold" size="20.0" />
                          </font>
                       </Text>
                       <Button fx:id="boton1" maxHeight="-Infinity"
maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
```

```
mnemonicParsing="false" prefHeight="60.0" prefWidth="180.0">
                           <font>
                              <Font name="Verdana Bold" size="37.0" />
                           </font>
                           <graphic>
                              <ImageView fx:id="imgBoton1"
fitHeight="50.0" fitWidth="50.0" pickOnBounds="true" preserveRatio="true">
                                 <image>
                                    <Image url="@../assets/icons/play.png"
/>
                                 </image>
                              </ImageView>
                           </graphic>
                        </Button>
                        <Text fx:id="textTitulo2" strokeType="OUTSIDE"
strokeWidth="0.0" text="Cargar laberinto" textAlignment="CENTER"
textOrigin="CENTER">
                           <font>
                              <Font name="Verdana Bold" size="20.0" />
                           </font>
                        </Text>
                        <Button fx:id="boton2" maxHeight="-Infinity"
maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
mnemonicParsing="false" prefHeight="60.0" prefWidth="180.0">
                           <graphic>
                              <ImageView fx:id="imgBoton2"
fitHeight="50.0" fitWidth="50.0" pickOnBounds="true" preserveRatio="true"
rotate="180.0">
                                 <image>
                                    <Image
url="@../assets/icons/upload.png" />
                                 </image>
                              </ImageView>
                           </graphic>
                        </Button>
                        <Text strokeType="OUTSIDE" strokeWidth="0.0"
text="Hecho por:">
                           <font>
                              <Font name="Verdana Bold" size="16.0" />
                           </font>
                        </Text>
                        <Text strokeType="OUTSIDE" strokeWidth="0.0"
text=" Carlos Saúl Paz Maldonado" wrappingWidth="190.0">
                           <font>
                              <Font name="Verdana Bold Italic" size="22.0"
/>
                           </font>
                        </Text>
                     </children>
                  </VBox>
```

```
                </children>
            </AnchorPane>
        </items>
      </SplitPane>
   </children>
</AnchorPane>
```

21. CSS Styles

```
22. .grid-pane {
        -fx-grid-lines-visible: true;
    }

    .grid-pane .grid-line {
        -fx-stroke: green; /* Cambia el color aquí */
        -fx-stroke-width: 5; /* Cambia el grosor aquí */
    }
```

23. Clase Main

```
package unam.fesaragon.estructuradatos;

import javafx.application.Application;
import javafx.stage.Stage;
import unam.fesaragon.estructuradatos.controladores.Laberinto;


public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        Laberinto laberinto = new Laberinto(10,10,0.1);
        laberinto.comenzar();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

CONCLUSIONES

Tengo que admitir que el reto más grande en este proyecto no fue el algoritmo de back tracking si no la implementación de JavaFX, es una de las primeras veces que hago proyectos con este recurso. Aunque hubo momentos de frustración pude encontrar soluciones a errores a través de foros, tutoriales, la documentación, entre otros. Sin embargo, he aprendido un poco mejor de este recurso, que no dudare en utilizarlo nuevamente en algún otro proyecto.

Igualmente es una de las primeras veces que intento implementar un patrón de diseño, específicamente el patrón MVC; menciono la palabra “intento”, porque, aunque haya separado de esa forma los recursos, mis clases, componentes, excepciones, etc. estoy seguro de que más de un error de organización si está presente en mi proyecto. Tal vez pueda ser atribuido por mi inexperiencia con este patrón, e inclusive el limite de tiempo para entregar este proyecto. Pero de alguna manera he aprendido cosas nuevas que no había utilizado antes.

He de decir que por falta de tiempo no pude solucionar el pintado de celdas cuando se regresa el camino del laberinto, en cambio solo colorea de rojo las celdas que no encontrara el camino, ya que previamente se ha encontrado el camino correcto en la lógica. Aún así intentare solucionarlo en una rama secundaria del proyecto.