

# **CARTA DE PRESENTACIÓN**

***Laberinto ERA***

**UNIVERSIDAD CENTRAL DEL ECUADOR**



**FACULTAD DE INGENIERÍA CIENCIAS FÍSICAS Y  
MATEMÁTICA**

**CARRERA DE SISTEMAS DE LA INFORMACIÓN**

**GRUPO DE DESARROLLO DE SOFTWARE**

**ALGORITMOS**

**ANDY LLUMIQUINGA**

**ROYER MORENO**

**ERICK DÍAZ**

**2020**

---

## Contenido

1	Introducción al sistema LaberintoERA .....	5
2	Requerimientos de Hardware y de Software .....	5
2.1	Requerimientos de hardware cliente.....	5
2.2	Requerimientos de software cliente.....	5
3	Diseño de clases.....	6
3.1	Diagrama de clases resumido.....	6
3.2	Lista de objetos.....	9
	Información de clases.....	10
3.2.1	Inicio .....	10
3.2.2	JuegoGanado .....	12
3.2.3	JuegoPerdido.....	12
3.2.4	LaberintoGUI .....	13
3.2.5	ManejoVentana.....	14
3.2.6	Practicar.....	15
3.2.7	Terminar.....	15
3.2.8	Ventana1 .....	16
3.2.9	Abajo.....	17
3.2.10	Arriba.....	17
3.2.11	Ayuda .....	18
3.2.12	Derecha.....	18
3.2.13	Fin.....	19
3.2.14	Finpractica .....	20
3.2.15	InformacionInicio .....	21
3.2.16	Izquierda.....	21
3.2.17	ValoresValido.....	22
3.2.18	BackTRacking .....	22

---

3.2.19	Cargar .....	23
3.2.20	Coordenada .....	23
3.2.21	Mazr .....	24
3.2.22	MazeGenerator .....	25
3.2.23	Node.....	27
3.2.24	Sonido .....	27
3.2.25	Laberinto.....	28
3.2.26	Personaje.....	29
3.3	Información de los Metodos .....	32
3.3.1	Métodos Inicio.....	32
3.3.2	Métodos de JuegoGanado .....	32
3.3.3	Métodos JuegoPerdido.....	32
3.3.4	Métodos LaberintoGUI.....	33
3.3.5	Métodos ManejoVentana .....	33
3.3.6	Métodos Practicar .....	33
3.3.7	Métodos Terminar .....	33
3.3.8	Métodos Ventana1 .....	33
3.3.9	Métodos Abajo .....	34
3.3.10	Métodos Arriba .....	34
3.3.11	Métodos Derecha .....	34
3.3.12	Métodos Izquierda .....	34
3.3.13	Métodos Fin .....	35
3.3.14	Métodos Finpractica.....	35
3.3.15	Métodos InformacionInicio.....	35
3.3.16	Métodos BackTracking.....	35
3.3.17	Métodos Cargar.....	35
3.3.18	Métodos Coordenada.....	36
3.3.19	Métodos Maze.....	36

---

3.3.20	Métodos MazeGenerator .....	36
3.3.21	Métodos Node .....	37
3.3.22	Métodos Sonido.....	37
3.3.23	Métodos Laberinto .....	37
3.3.24	Métodos Personaje.....	37
3.4	Información de Eventos .....	39
3.4.1	Eventos JuegoGanado .....	39
3.4.2	Eventos JuegoPerdido.....	39
3.4.3	Eventos Practicar .....	39
3.4.4	Eventos Terminar .....	39
3.4.5	Eventos Ventana1 .....	39
3.4.6	Eventos Abajo.....	40
3.4.7	Eventos Arriba .....	40
3.4.8	Eventos Derecha.....	40
3.4.9	Eventos Izquierda.....	40
3.4.10	Eventos Inicio.....	40
3.4.11	Eventos Fin.....	40
3.4.12	Eventos FinPractica.....	40
4	Librerías programadas .....	40
4.1	Librería Forms .....	40
4.1.1	Lista de Objetos .....	41
5	Anexos.....	64

---

# MANUAL TÉCNICO

## 1 Introducción al sistema LaberintoERA

Este documento servirá a modo manual para poder entender el sistema LaberintoERA a profundidad. Este explicara cuales son los requisitos mínimos que requerirá el sistema para poder ejecutar el programa, como también se explicara mediante un diagrama clases y otro apartado más, los diferentes métodos, atributos, clases y a su vez de como estos elementos se interrelacionan entre sí. Todo esto con el fin de dejar un registro pertinente para que una persona ajena o no al desarrollo de producto pueda saber cómo funciona y pueda modificarlo ya sea para satisfacer una necesidad particular o para mejorar el programa en sí. Tener en cuenta que dicho sistema será libre y sin ningún tipo afirmación que diga lo contrario para poder ser replicado u modificado, una vez subido a GitHub cualquier apasionado en los sistemas como este y desarrollador en lenguaje Java podrá acceder.

## 2 Requerimientos de Hardware y de Software

### 2.1 Requerimientos de hardware cliente

Procesador: Intel Core i5-5200U de 64bits CPU @ 2.20 Ghz

Memoria RAM: 1GB DDR3 1200Hz

Tarjeta Gráfica: Intel HUD graphics

### 2.2 Requerimientos de software cliente

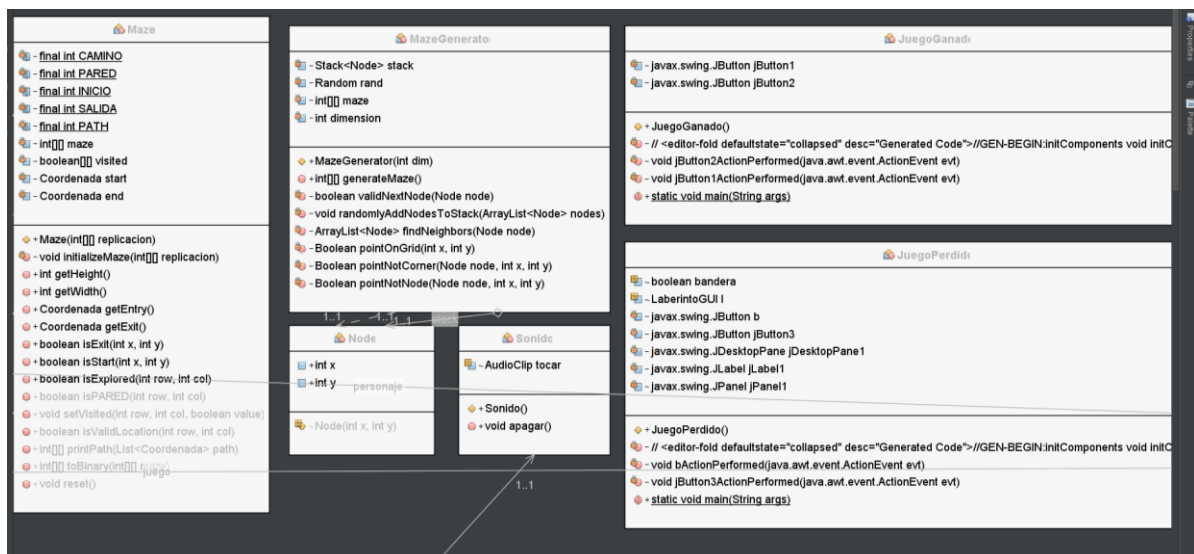
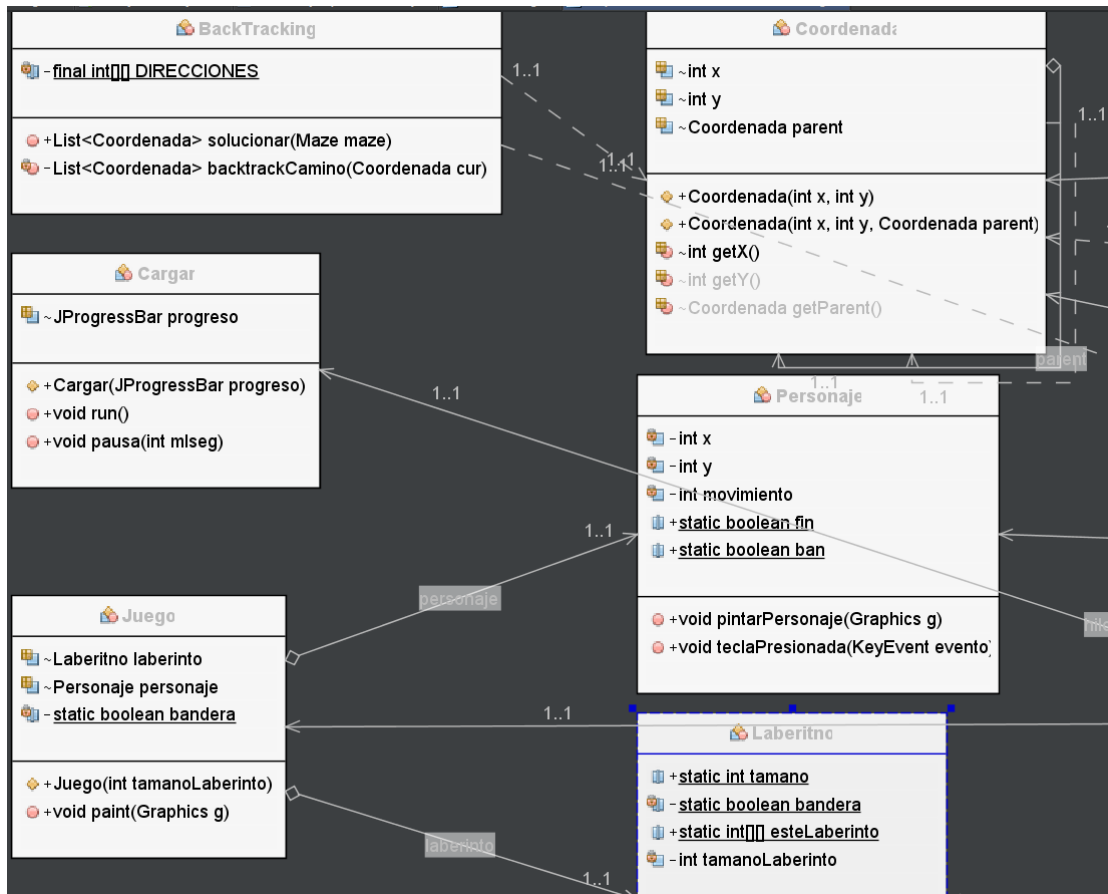
Sistema Operativo: Microsoft Windows 10 Home Single Language

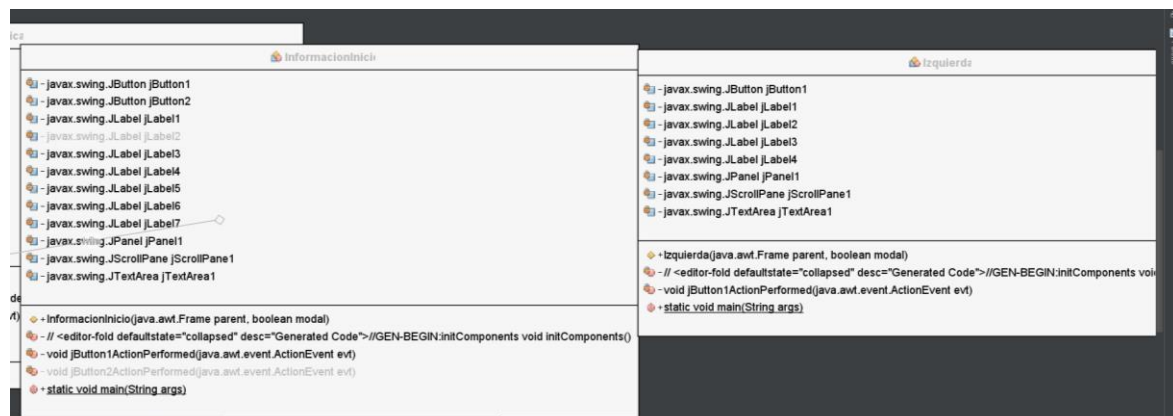
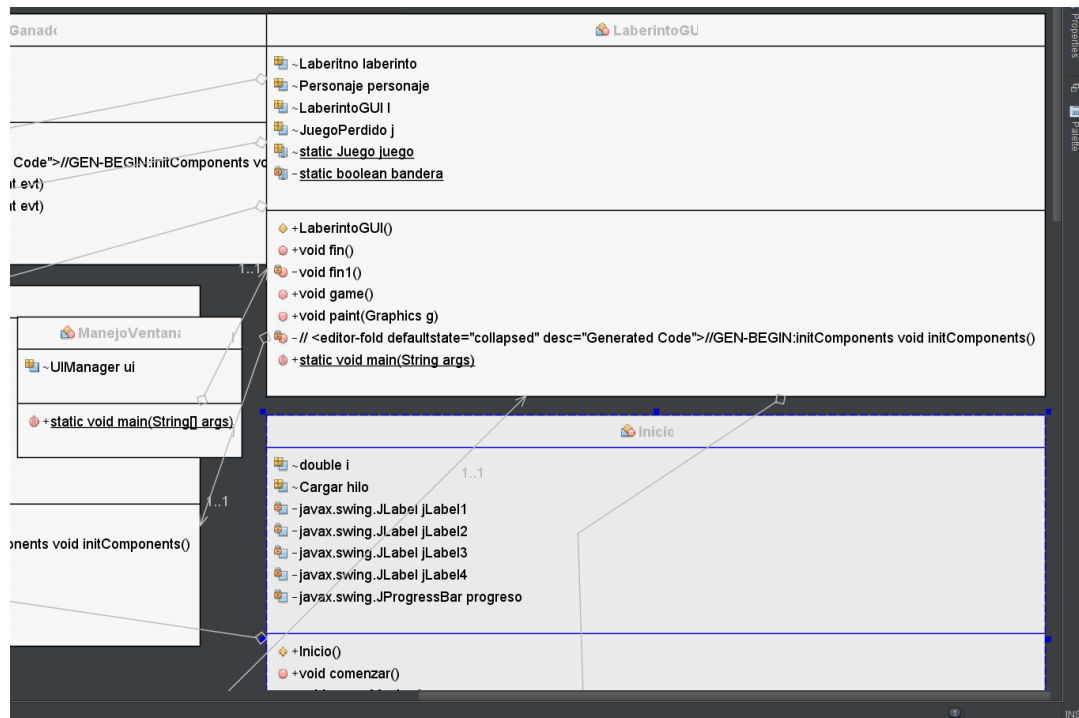
Java JDK Version 8 Update 1.8.0\_261

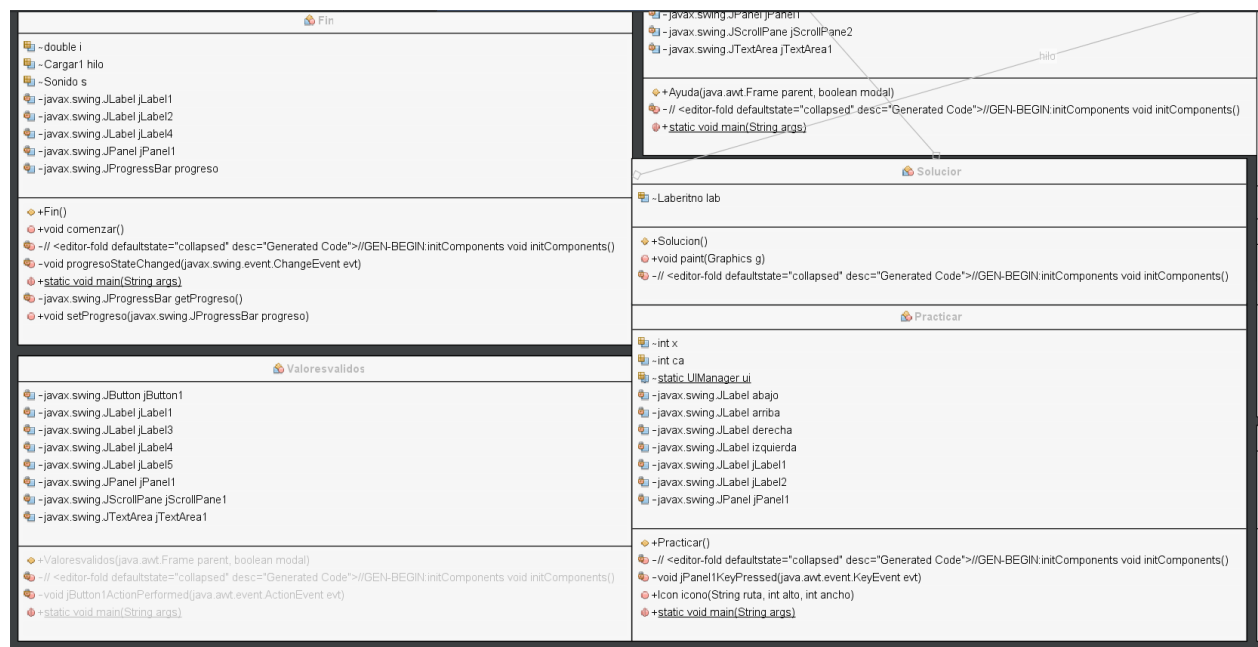
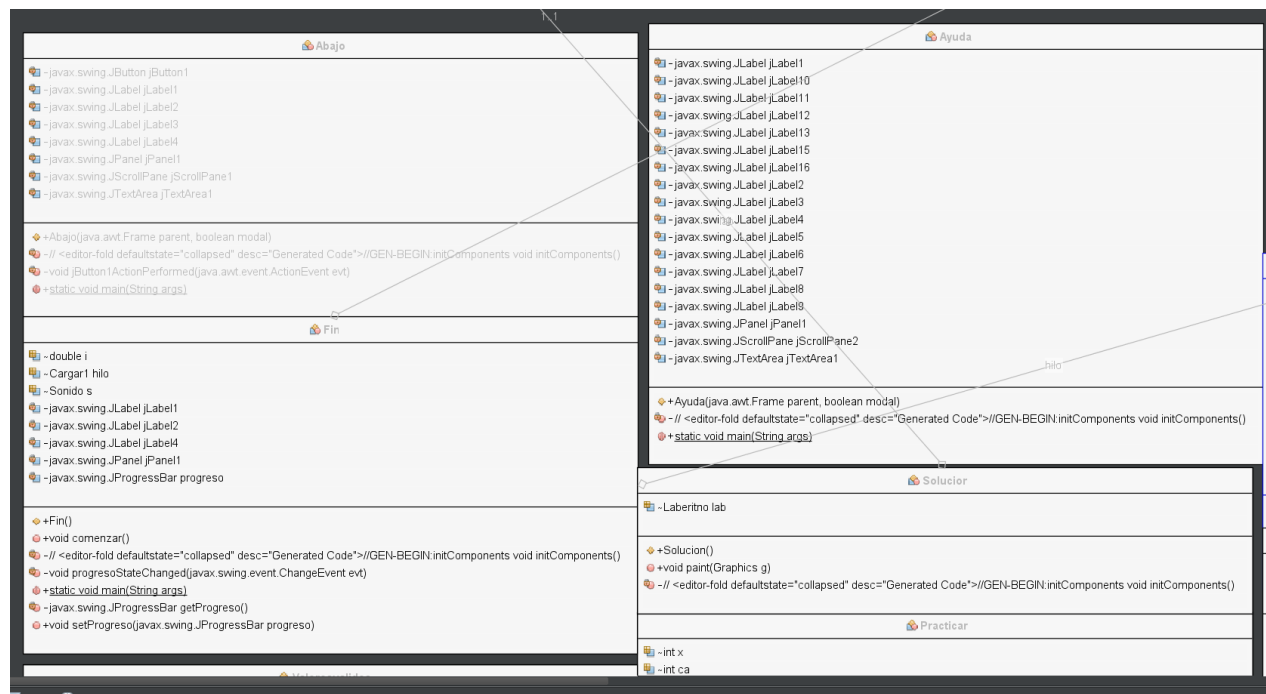
JRE Version 1.8.0\_261

### 3 Diseño de clases.

#### 3.1 Diagrama de clases resumido




























<pre> - javax.swing.JLabel jLabel5 - javax.swing.JLabel jLabel6 - javax.swing.JPanel jPanel1 - javax.swing.JScrollPane jScrollPane1 - javax.swing.JTextArea jTextArea1  +Finpractica(java.awt.Frame parent, boolean modal) - // &lt;editor-fold defaultstate="collapsed" desc="Generated Code"&gt; //GEN-BEGIN: initComponents void initComponents() - void jButton1ActionPerformed(java.awt.event.ActionEvent evt) - +static void main(String args) </pre>	
<p>Terminar</p> <pre> - Timer t - AudioClip sonar  +Terminar() +void actionPerformed(ActionEvent e) +void musica() - // &lt;editor-fold defaultstate="collapsed" desc="Generated Code"&gt; //GEN-BEGIN: initComponents void initComponents() - +static void main(String args) </pre>	
<p>Ventana2</p> <pre> - javax.swing.JButton cerrar - javax.swing.JLabel datos - javax.swing.JPasswordField password  +Ventana2() +void cerrar() - // &lt;editor-fold defaultstate="collapsed" desc="Generated Code"&gt; //GEN-BEGIN: initComponents void initComponents() - void cerrarActionPerformed(java.awt.event.ActionEvent evt) - +static void main(String args) </pre>	<p>Ventana1</p> <pre> +static int tamanolaberinto - javax.swing.JButton jButton1 - javax.swing.JLabel jLabel1 - javax.swing.JLabel jLabel2 - javax.swing.JMenu jMenu1 - javax.swing.JMenuBar jMenuBar1 - javax.swing.JMenuItem jMenuItem1 - javax.swing.JMenuItem jMenuItem2 - javax.swing.JMenuItem jMenuItem3 - javax.swing.JMenuItem jMenuItem4 - javax.swing.JMenuItem jMenuItem5 - javax.swing.JMenuItem jMenuItem6 - javax.swing.JPopupMenu.Separator jSeparator2 - javax.swing.JTextField nombre  +Ventana1() - // &lt;editor-fold defaultstate="collapsed" desc="Generated Code"&gt; //GEN-BEGIN: initComponents void initComponents() - void jButton1ActionPerformed(java.awt.event.ActionEvent evt) - void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) - void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) - void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) - void nombreKeyTyped(java.awt.event.KeyEvent evt) - void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) - void nombreKeyPressed(java.awt.event.KeyEvent evt) - void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) - void jButton1KeyPressed(java.awt.event.KeyEvent evt) - +static void main(String args) </pre>














## 3.2 Lista de objetos

### Lista de forms

-  Inicio
-  JuegoGanado
-  JuegoPedido
-  LaberintoGUI
-  ManejoVentana
-  Practicar
-  Terminar
-  Ventana1
-  Abajo
-  Arriba
-  Ayuda
-  Derecha
-  Fin
-  FinPractica

-  InformacionInicio
-  Izquierda
-  Valoresvalidos
-  Log
-  Registrar

## Lista de clases

-  BackTracking
-  Sondio
-  Cargar
-  Coordenanda
-  Maze
-  MazeGenerator
-  Node
-  Juego
-  Personaje
-  Laberinto
-  DataBase
-  Usuario
-  UsuarioTrs

## Información de clases

### 3.2.1 Inicio

#### 3.2.1.1 Parámetros

La clase Inicio no tiene parámetros

### 3.2.1.2 Atributos

double i

double j

Cargar hilo

Sonido s

AudioClip sound

private JLabel1

private JLabel2

private JLabel3

private JLabel4

pivate JProgressBar progreso

### 3.2.1.3 Métodos

public Inicio()

public void comenzar()

public void cargarMusica()

private void progresoStateChanged(javax.swing.event.ChangeEvent evt)

public Icon icono(String ruta, int alto, int ancho)

public static void main(String args[])

### 3.2.1.4 Finalidad

La clase Inicio tiene como finalidad de que una vez el usuario ingrese al juego, aparezca una pantalla de bienvenida la cual detalla una forma de cómo será el juego, la pantalla estará activa durante cierto periodo de tiempo, posterior seguirá una pantalla de tutorial

### 3.2.1.5 Herencia

javax.swing.JFrame

## **3.2.2 JuegoGanado**

### **3.2.2.1 Parámetros**

La clase registro no tiene ningún parámetro

### **3.2.2.2 Atributos**

AudioClip sound

### **3.2.2.3 Métodos**

```
public JuegoGanado()
```

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
```

```
public static void main(String args[])
```

### **3.2.2.4 Finalidad**

Esta ventana cuenta con la finalidad de que una vez el personaje dentro del laberinto llega hacia la meta, saldrá una pantalla felicitando el trabajo que hace y si es que desea intentar una nueva partida o si desde salir del juego.

### **3.2.2.5 Herencia**

javax.swing.JFrame

## **3.2.3 JuegoPerdido**

### **3.2.3.1 Parámetros**

Esta clase no tiene ningún parámetro

### **3.2.3.2 Atributos**

Boolean bandera

AudioClip sound

LaberintoGUI I

### 3.2.3.3 Metodos

```
public JuegoPerdido()
```

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
```

```
public static void main(String args[])
```

### 3.2.3.4 Finalidad

Este JFrameForm tiene la finalidad de que una vez el usuario haya presionado la tecla Y para observar la solución del laberinto entonces, se considera a ese intento como perdido y deberá de entrar en nuevamente al programar, esto es pensado de esta manera pues las máquina con monedas, comúnmente usan esta técnica en donde si desea intentar nuevamente el juego debe de ingresar una nueva moneda.

### 3.2.3.5 Herencia

```
javax.swing.JFrame
```

## 3.2.4 LaberintoGUI

### 3.2.4.1 Parametros

Esta clase no cuenta con ningún parámetro

### 3.2.4.2 Atributos

```
JButton cerrar
```

```
JLabel datos
```

### 3.2.4.3 Metodos

```
public Ventana2()
```

```
public void cerrar()
```

```
private void cerrarActionPerformed(java.awt.event.ActionEvent evt)
```

```
public static void main(String args[])
```

#### 3.2.4.4 Finalidad

Tiene la finalidad de indicar una animación de fin de juego una vez que el laberinto a sido completado ya sea porque ha ganado por que ha perdido, de igual forma está validado para que cuando desee salir aparezca.

#### 3.2.4.5 Herencia

javax.swing.JFrame

### 3.2.5 ManejoVentana

#### 3.2.5.1 Parámetros

La clase ManejoVentana no tiene ningún parámetro.

#### 3.2.5.2 Atributos

UIManager ui

```
Player rep = new Player(new FileInputStream("C:\\Users\\Andy\\Desktop\\UCE\\SEMESTRE 2020-2020 4\\Algoritmos\\Proyecto\\ProyectoLaberinto\\src\\sonido\\sonid.mp3"));
```

#### 3.2.5.3 Métodos

```
public static void main(String[] args) throws FileNotFoundException, JavaLayerException
```

```
    rep.play();
```

#### 3.2.5.4 Finalidad

La Ventana ManejoVentana será la encargada de iniciar al programa, esta es la pantalla donde nuestra clase Inicio va a iniciar en, se hizo uso de clases como Player que contiene métodos para que el sonido pueda ser reproducido.

#### 3.2.5.5 Herencia

Esta clase no tiene ninguna herencia

### **3.2.6 Practicar**

#### **3.2.6.1 Parametros**

Esta clase no tiene ningún parametro

#### **3.2.6.2 Atributos**

Int x

Int y

Int ca, cb, ci, cd

UIManager ui

#### **3.2.6.3 Metodos**

```
public Practicar()
```

```
private void jPanel1KeyPressed(java.awt.event.KeyEvent evt)
```

```
public Icon icono(String ruta, int alto, int ancho)
```

```
public static void main(String args[])
```

#### **3.2.6.4 Finalidad**

Esta clase tiene la finalidad de presentar un tutorial de como el jugador manejara al personaje dentro del juego, debe de aplastar las teclas de direcciones hasta el fondo de la ventana para cumplir con el tutorial

#### **3.2.6.5 Herencia**

javax.swing.JFrame

### **3.2.7 Terminar**

#### **3.2.7.1 Parametros**

Esta clase no tiene ningún parametro

### 3.2.7.2 Atributos

Timer t

AudioClip sonar

### 3.2.7.3 Metodos

```
public void actionPerformed(ActionEvent e)
```

```
public void musica()
```

```
public static void main(String args[])
```

### 3.2.7.4 Finalidad

Esta clase tiene la finalidad de presentar un soundtrack cuando el juego termine

### 3.2.7.5 Herencia

javax.swing.JFrame

## 3.2.8 Ventana1

### 3.2.8.1 Parámetros

La clase Ventana1 no tiene ningún parámetro

### 3.2.8.2 Atributos

```
public static int tamanoLaberitno;
```

### 3.2.8.3 Métodos

```
public Ventana1()
```

### 3.2.8.4 Finalidad

Esta clase servirá como pantalla para iniciar el tamaño del laberinto, ver los diferentes apartados de opciones que se tiene para obtener información acerca del juego o reiniciarlo enviará el tamaño necesario para generar el laberinto de camino aleatorio



### **3.2.8.5 Herencia**

javax.swing.JFrame

### **3.2.9 Abajo**

#### **3.2.9.1 Parametros**

Esta clase no posee ningún parametro

#### **3.2.9.2 Atributos**

Esta clase no posee ningún atributo

#### **3.2.9.3 Metodos**

```
public Abajo(java.awt.Frame parent, boolean modal)
```

```
public static void main(String args[])
```

#### **3.2.9.4 Finalidad**

Esta clase tiene la finalidad de mostrar una retroalimentación a la hora de presionar la tecla hacia abajo en el juego, una vez alcanzada el borde de la ventana de práctica, esta ventana saldrá de manera automática

### **3.2.9.5 Herencia**

javax.swing.JDialog

### **3.2.10 Arriba**

#### **3.2.10.1 Parametros**

Esta clase no cuenta con parametros

#### **3.2.10.2 Atributos**

Esta clase no cuenta con atributos

#### **3.2.10.3 Metodos**

```
public Arriba(java.awt.Frame parent, boolean modal)
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
```

```
public static void main(String args[])
```

#### **3.2.10.4 Finalidad**

Esta clase tiene la finalidad de mostrar una retroalimentación a la hora de presionar la tecla hacia arriba en el juego, una vez alcanzada el borde de la ventana de práctica, esta ventana saldrá de manera automática

#### **3.2.10.5 Herencia**

```
javax.swing.JDialog
```

### **3.2.11 Ayuda**

#### **3.2.11.1 Parametros**

Esta clase no implemente ningún parametro

#### **3.2.11.2 Atributos**

Esta clase no implemente ningún atributo

#### **3.2.11.3 Metodos**

```
public Ayuda(java.awt.Frame parent, boolean modal)
```

```
public static void main(String args[])
```

#### **3.2.11.4 Finalidad**

Esta clase tiene la finalidad de mostrar los diferentes objetivos y recomendaciones cuando entre al juego, le permite entender si es que en algún momento no entiende que debe de hacer o que prosigue en la actual etapa, si el usuario necesita ayuda se muestra como una opción en la barra de menú

#### **3.2.11.5 Herencia**

```
javax.swing.JDialog
```

### **3.2.12 Derecha**

#### **3.2.12.1 Parametros**

Esta clase no cuenta con parametros

### **3.2.12.2 Atributos**

Esta clase no cuenta con atributos

### **3.2.12.3 Metodos**

```
public Derecha(java.awt.Frame parent, boolean modal)
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
```

### **3.2.12.4 Finalidad**

Esta clase tiene la finalidad de mostrar una retroalimentación a la hora de presionar la tecla hacia la derecha en el juego, una vez alcanzada el borde de la ventana de práctica, esta ventana saldrá de manera automática.

### **3.2.12.5 Herencia**

```
javax.swing.JDialog
```

## **3.2.13 Fin**

### **3.2.13.1 Parametros**

Esta clase no tiene ningún parámetro

### **3.2.13.2 Atributos**

Double i

Double j

Cargar1 hilo

Sonido s

### **3.2.13.3 Métodos**

```
public Fin()
```

```
public void comenzar()
```

```
public void cargarMusica()
```

```
private void progresoStateChanged(javax.swing.event.ChangeEvent evt)

private javax.swing.JProgressBar getProgreso()

public void setProgreso(javax.swing.JProgressBar progreso)

public static void main(String[] args)
```

#### **3.2.13.4 Finalidad**

Tiene la finalidad de finalizar el programa, con un soundtrack diferente al que se le había realizado

#### **3.2.13.5 Herencia**

javax.swing.JDialog

#### **3.2.14 Finpractica**

##### **3.2.14.1 Parametros**

Esta clase no cuenta con parametros

##### **3.2.14.2 Atributos**

Esta clase no cuenta con atributos

##### **3.2.14.3 Métodos**

```
public Finpractica(java.awt.Frame parent, boolean modal)

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)

public static void main(String args[])
```

##### **3.2.14.4 Finalidad**

Esta clase tiene como finalidad el que una vez terminada la practica aparece un mensaje que le permite seguir a la pantalla de juego para iniciar el laberinto

##### **3.2.14.5 Herencia**

javax.swing.JDialog

### **3.2.15 InformacionInicio**

#### **3.2.15.1 Parametros**

No cuenta con parametros

#### **3.2.15.2 Atributos**

No cuenta con atributos

#### **3.2.15.3 Métodos**

```
public InformacionInicio(java.awt.Frame parent, boolean modal)

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)

public static void main(String args[])
```

#### **3.2.15.4 Finalidad**

Esta clase tiene como finalidad el de iniciar una introducción de lo que va a tratar el tutorial inicial el cual debe de terminarlo para proseguir

#### **3.2.15.5 Herencia**

javax.swing.JDialog

### **3.2.16 Izquierda**

#### **3.2.16.1 Parametros**

No tiene parametros

#### **3.2.16.2 Atributos**

No cuenta con atributos

#### **3.2.16.3 Métodos**

```
public Izquierda(java.awt.Frame parent, boolean modal)

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)

public static void main(String args[])
```

### **3.2.16.4 Finalidad**

Esta clase tiene la finalidad de mostrar una retroalimentación a la hora de presionar la tecla hacia la izquierda en el juego, una vez alcanzada el borde de la ventana de práctica, esta ventana saldrá de manera automática.

### **3.2.17 ValoresValido**

#### **3.2.17.1 Parametros**

No cuenta con parametros

#### **3.2.17.2 Atributos**

No cuenta con atributos

#### **3.2.17.3 Métodos**

```
public Valoresvalidos(java.awt.Frame parent, boolean modal)
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
```

```
public static void main(String args[])
```

#### **3.2.17.4 Finalidad**

Tiene la finalidad de indicar si es que los campos ingresados para la dimensión del laberinto es correcta o no.

### **3.2.18 BackTRacking**

#### **3.2.18.1 Parametros**

```
int[][] DIRECCIONES
```

#### **3.2.18.2 Atributos**

```
private static final int[][] DIRECCIONES = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
```

#### **3.2.18.3 Métodos**

```
public List<Coordenada> solucionar(Maze maze)
```

```
private List<Coordenada> backtrackCamino(Coordenada cur)
```

### **3.2.18.4 Finalidad**

Tiene la finalidad de solucionar el laberinto que se le asigne y posterior trazar el camino solución

### **3.2.19 Cargar**

#### **3.2.19.1 Parametros**

No tiene parametros

#### **3.2.19.2 Atributos**

JProgressBar progreso;

#### **3.2.19.3 Métodos**

```
public Cargar(JProgressBar progreso)
```

```
public void run()
```

```
public void pausa(int mlseg)
```

#### **3.2.19.4 Finalidad**

Tiene la finalidad de ayudar con los procesos relacionados a hilos, para aquellas clases como inicio que muestran los procesos que se van a realizar

#### **3.2.19.5 Herencia**

Thread

### **3.2.20 Coordinada**

#### **3.2.20.1 Parametros**

Int x

Int y

Coordinada parent;

#### **3.2.20.2 Atributos**

int x;

int y;

Coordenada parent;

### **3.2.20.3 Métodos**

```
public Coordenada(int x, int y)
```

```
public Coordenada(int x, int y, Coordenada parent)
```

### **3.2.20.4 Finalidad**

Tiene la finalidad de ayudar a la clase que solucionada el laberinto usando las coordenadas para la ubicación dentro del laberinto

### **3.2.20.5 Herencia**

No tiene herencia

## **3.2.21 Mazr**

### **3.2.21.1 Parametros**

int CAMINO

int PARED

int INICIO

int SALIDA

int PATH

int[][] maze

boolean[][] visited

Coordenada start

Coordenada end

### **3.2.21.2 Atributos**

```
private static final int CAMINO = 0;
```

```
private static final int PARED = 1;
```

```
private static final int INICIO = 2;
```

```
private static final int SALIDA = 3;
```



```
private static final int PATH = 4;

private int[][] maze;

private boolean[][] visited;

private Coordenada start;

private Coordenada end;
```

### 3.2.21.3 Métodos

```
public Maze(int[][] replicacion)

private void initializeMaze(int[][] replicacion)

public boolean isExit(int x, int y)

public boolean isStart(int x, int y)

public boolean isExplored(int row, int col)

public boolean isPARED(int row, int col)

public void setVisited(int row, int col, boolean value)

public boolean isValidLocation(int row, int col)

public int[][] printPath(List<Coordenada> path)

public int[][] toBinary(int[][] maze)

public void reset()
```

### 3.2.21.4 Finalidad

Tiene la finalidad de transformar el laberinto que se le otorga a una estructura en la que el solucionador del laberinto mediante backtracking va a aceptar, además de obtener tanto las entradas, salidas, paredes y caminos como métodos que devuelven el valor de los atributos ya definidos anteriormente

### 3.2.21.5 Herencia

No hereda de otra clase

## 3.2.22 MazeGenerator

### 3.2.22.1 Parametros

```
Stack<Node> stack = new Stack<>()
```

```
Random rand = new Random()
```

```
int[][] maze
```

```
int dimension
```

### **3.2.22.2 Atributos**

```
private Stack<Node> stack = new Stack<>();
```

```
private Random rand = new Random();
```

```
private int[][] maze;
```

```
private int dimension;
```

### **3.2.22.3 Métodos**

```
public MazeGenerator(int dim)
```

```
public int[][] generateMaze()
```

```
private boolean validNextNode(Node node)
```

```
private void randomlyAddNodesToStack(ArrayList<Node> nodes)
```

```
private ArrayList<Node> findNeighbors(Node node)
```

```
private Boolean pointOnGrid(int x, int y)
```

```
private Boolean pointNotCorner(Node node, int x, int y)
```

```
private Boolean pointNotNode(Node node, int x, int y)
```

### **3.2.22.4 Finalidad**

Esta clase tiene la finalidad de generar un laberinto dependiendo de la dimensión que se le conceda al mismo, además va a realizar la estructura de manera que exista siempre una salida válida y con una única solución, debe de ser capaz de identificar esto usando las posiciones de los vecinos y que además sepa cómo se va a ir generando la posible solución

### **3.2.22.5 Herencia**

No tiene herencia

### 3.2.23 Node

#### 3.2.23.1 Parametros

int x

int y

#### 3.2.23.2 Atributos

public final int x;

public final int y;

#### 3.2.23.3 Métodos

Node(int x, int y)

#### 3.2.23.4 Finalidad

Tiene la finalidad de crear coordenadas para el personaje tanto en x como y

#### 3.2.23.5 Herencia

No tiene herencia

### 3.2.24 Sonido

#### 3.2.24.1 Parametros

AudioClip tocar

#### 3.2.24.2 Atributos

AudioClip tocar;

#### 3.2.24.3 Métodos

public Sonido()

public void apagar()

#### 3.2.24.4 Finalidad

Esta clase tiene la finalidad de cargar el soundtrack deseado para el juego

### **3.2.24.5 Herencia**

No tiene herencia

### **3.2.25 Laberinto**

#### **3.2.25.1 Parametros**

int tamaño

boolean bandera = true

int[][] esteLaberinto

int tamanoLaberinto

#### **3.2.25.2 Atributos**

```
public static int tamano=0;
```

```
    private static boolean bandera = true;
```

```
    public static int[][] esteLaberinto;
```

```
    private int tamanoLaberinto;
```

#### **3.2.25.3 Métodos**

```
public Laberitno(int tamanoLaberinto)
```

```
public void teclaPresionada(KeyEvent evento)
```

```
public void pintarLaberinto(Graphics g)
```

```
public int[][] obtenerLaberinto()
```

#### **3.2.25.4 Finalidad**

Esta clase tiene la finalidad de devolver y determinar los tamaños de la ventana que van a hacer uso de esta clase para dibujar al laberinto, de igual forma permitirá determinar el camino solución del laberinto una vez que se haya indicado el comando correspondiente para la solución desde aquí se va a manejar lo correspondiente a creación de los laberintos.

#### **3.2.25.5 Herencia**

Thread

### 3.2.26 Personaje

#### 3.2.26.1 Parametros

int x

int y

int movimiento

boolean fin

#### 3.2.26.2 Atributos

```
private int x = Laberitno.tamano;
```

```
    private int y = Laberitno.tamano;
```

```
    private int movimiento = 0;
```

```
    public static boolean fin;
```

#### 3.2.26.3 Métodos

```
public void pintarPersonaje(Graphics g)
```

```
public void teclaPresionada(KeyEvent evento)
```

#### 3.2.26.4 Finalidad

Tiene la finalidad de crear al personaje dentro del laberinto, el cual será dibujado mediante coordenadas tanto x como y, y sus posiciones serán tomadas dependiendo del tamaño del laberinto que se haya generado, también permite que solo transite por los lugares donde hay camino y no paredes.

#### 3.2.26.5 Herencia

Thread

### 3.2.27 Database

#### 3.2.27.1 Parametros

```
String path = System.getProperty("user.home") + "/Desktop/Database";
```

#### 3.2.27.2 Atributos

```
List<T> listObjects
```

String file

File thisFile, directory

### **3.2.27.3 Métodos**

DataBase(String nameFile)

createFile(String nameFile)

void writeFile()

void readFile()

abstract void tDefault()

### **3.2.27.4 Finalidad**

Es la clase que permite realizar operaciones de administración para agregar usuarios en un fichero binario para poder registrarse dentro del juego y puedan acceder a la aplicación, la clase es abstracta pues todos los métodos de inserción se harán desde la clase creada para el objeto a controlar.

### **3.2.27.5 Herencia**

No cuenta con herencia

## **3.2.28 Usuario**

### **3.2.28.1 Parametros**

final long serialVersionUID = -1L

### **3.2.28.2 Atributos**

String nombreUsuario

String cotrasena

### **3.2.28.3 Métodos**

Usuario(String nombreUsuario, String cotrasena)

String getNombreUsuario()

setNombreUsuario(String nombreUsuario)

String getCotrasena()

setCotrasena(String cotrasena)

### **3.2.28.4 Herencia**

Implements Serializable

### **3.2.29 UsuarioTrs**

#### **3.2.29.1 Parametros**

No cuenta con parametros

#### **3.2.29.2 Atributos**

No cuenta con atributos

#### **3.2.29.3 Métodos**

String crearUsuario(Usuario registro)

void tDefault()

boolean buscarUsuario(String usuario)

List<Usuario> read()

#### **3.2.29.4 Finalidad**

La finalidad de este es realizar todas las operaciones de creación que serán destinadas a ser guardadas dentro de los archivos binarios. Además de devolver todos los datos que se almacenen en un arrayList para determinar que usuarios existen.

#### **3.2.29.5 Herencia**

Database<Usuario>

### **3.2.30 Log**

#### **3.2.30.1 Parametros**

No cuenta con parametros

#### **3.2.30.2 Atributos**

No cuenta con atributos

#### **3.2.30.3 Métodos**

public Log()

```
void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
```

```
void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
```

```
static void main(String args[]
```

### 3.2.30.4 Finalidad

Tiene la finalidad de mostrar todos los campos a ser completados para poner iniciar sesión, también redirige a la opción de poder ir a una ventana donde se pueda registrar el usuario nuevo.

### 3.2.30.5 Herencia

jFrame

## 3.3 Información de los Metodos

### 3.3.1 Métodos Inicio

Lista de métodos de la clase con sus descripciones:

- public Inicio()  
Encargado de inicializar todos los componentes del frame
- public void comenzar()  
Una vez terminada la presentación dará inicio a la carga de tutorial
- public void cargarMusica()
- private void progresoStateChanged(javax.swing.event.ChangeEvent evt)  
Determina cuanto tiempo se va a realizar la presentación de bienvenida
- public Icon icono(String ruta, int alto, int ancho)  
Carga un icono en el frame
- public static void main(String args[])  
Método que iniciará la creación del JFrameForm y lo hará visible.

### 3.3.2 Métodos de JuegoGanado

- public JuegoGanado()  
Constructor que permite generar el frame, además de poner icono a la ventana, parámetros que permitirán que el frame se coloque en medio de la pantalla de manera centrada
- public static void main(String args[])  
Método que iniciará la creación del JFrameForm y lo hará visible.

### 3.3.3 Métodos JuegoPerdido

- public JuegoPerdido()  
Constructor que permite generar el frame, además de poner icono a la ventana, parámetros que permitirán que el frame se coloque en medio de la pantalla de manera centrada
- public static void main(String args[])



---

Método que iniciará la creación del JFrameForm y lo hará visible.

### 3.3.4 Métodos LaberintoGUI

- `public LaberintoGUI()`  
Método que permite dibujar el laberinto una vez se le a proporcionado un tamaño para el mismo además de encargado de inicializa los componentes del JFrameForm
- `public static void main(String args[])`  
Método que iniciara la creación del JFrameForm y lo hará visible.

### 3.3.5 Métodos ManejoVentana

- `public static void main(String args[])`  
Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.6 Métodos Practicar

- `public Practicar()`  
Método encargado de inicializa los componentes del JFrameForm
- `public static void main(String args[])`  
Método que iniciara la creación del JFrameForm y lo hará visible
- `public Icon icono(String ruta, int alto, int ancho)`  
Método que se encargara de realizar la incorporación de los íconos en las diferentes pantallas

### 3.3.7 Métodos Terminar

- `public Terminar()`  
Método encargado de inicializa los componentes del JFrameForm
- `public void musica()`  
Método que permite inicializar el soundtrack dentro del juego
- `public static void main(String args[])`  
Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.8 Métodos Ventana1

- `public Ventana1()`  
Método encargado de inicializa los componentes del JFrameForm
- `public static void main(String args[])`  
Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.9 Métodos Abajo

- `public Abajo(java.awt.Frame parent, boolean modal)`

Método que permite inicializar y validar que el tutorial donde se hace uso de las flechas se complete correctamente, hay que tener en cuenta que para superar el tutorial se debe de llegar hasta el borde de la ventana

- `public static void main(String args[])`

Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.10 Métodos Arriba

- `public Arriba(java.awt.Frame parent, boolean modal)`

Método que permite inicializar y validar que el tutorial donde se hace uso de las flechas se complete correctamente, hay que tener en cuenta que para superar el tutorial se debe de llegar hasta el borde de la ventana

- `public static void main(String args[])`

Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.11 Métodos Derecha

- `public Derecha(java.awt.Frame parent, boolean modal)`

Método que permite inicializar y validar que el tutorial donde se hace uso de las flechas se complete correctamente, hay que tener en cuenta que para superar el tutorial se debe de llegar hasta el borde de la ventana

- `public static void main(String args[])`

Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.12 Métodos Izquierda

- `public Izquierda(java.awt.Frame parent, boolean modal)`

Método que permite inicializar y validar que el tutorial donde se hace uso de las flechas se complete correctamente, hay que tener en cuenta que para superar el tutorial se debe de llegar hasta el borde de la ventana

- `public static void main(String args[])`  
Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.13 Métodos Fin

- `public Fin()`  
Método que permite observar una animación una vez termine el juego.
- `public void comenzar()`  
Si el juego está iniciando entonces mandará a llamar a la correspondiente ventana que permitirá observar el tutorial del juego
- `public void cargarMusica()`  
Permite cargar música dentro del programa y que se la pueda manejar en las diferentes etapas del videojuego

### 3.3.14 Métodos Finpractica

- `public Finpractica(java.awt.Frame parent, boolean modal)`  
Método encargado de inicializa los componentes del JFrameForm
- `public static void main(String args[])`  
Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.15 Métodos InformacionInicio

- `public InformacionInicio(java.awt.Frame parent, boolean modal)`  
Método encargado de inicializa los componentes del JFrameForm
- `public static void main(String args[])`  
Método que iniciara la creación del JFrameForm inicial y lo hará visible

### 3.3.16 Métodos BackTracking

- `public List<Coordenada> solucionar(Maze maze)`  
Método que permite solucionar el laberinto que se le envía como parámetro y devolverá una vez encontrada la solución una lista con parámetros para crear la ruta de solución
- `private List<Coordenada> backtrackCamino(Coordenada cur)`  
Método que ayudará a crear las coordenadas que el método solucionar necesitará, las coordenadas se explican en otra clase de cómo funcionan sus atributos

### 3.3.17 Métodos Cargar

- `public Cargar(JProgressBar progreso)`  
Funciona como ayuda para cargar la pantalla de bienvenida a través de contadores y uso de hilos
- `public void run()`  
Usado como soporte para que la pantalla de carga se la pueda aplicar el efecto de desvanecido, estos métodos se los reutiliza para otras ventanas
- `public void pausa(int mseg)`

Método realiza los procesos de parar los hilos en un instante determinado en este caso será luego de nmseg

### 3.3.18 Métodos Coordenada

- `public Coordenada(int x, int y)`  
Método constructor de la clase coordenada que permite generar dichos valores dependiendo del flujo de datos que se obtengan en la solución del laberinto, este otro método está ubicado en la clase backtracking denominado como solución.
- `public Coordenada(int x, int y, Coordenada parent)`  
Método constructor que determinará si la coordenada obtenida corresponde a una coordenada factible haciendo uso de su coordenada padre, esto ayuda a determinar como la coordenada sí corresponderá a una solución.

### 3.3.19 Métodos Maze

- `public Maze(int[][] replicacion)`  
Con ayuda de este método constructor permite que una vez se ingrese la matriz o el laberinto en este caso, la transforme en una estructura tratable para solucionarla en este caso se definen tanto la pared como el camino además del inicio y final para formar la nueva matriz.
- `private void initializeMaze(int[][] replicacion)`  
Aquí se realiza todo el proceso descrito anteriormente
- `public boolean isExit(int x, int y)`  
Método que determina si la coordenada obtenida para formar el camino solución corresponde a una salida
- `public boolean isStart(int x, int y)`  
Método que determina si la coordenada obtenida para formar el camino solución corresponde a una entrada
- `public boolean isExplored(int row, int col)`  
Método que determina si la coordenada obtenida para formar el camino solución corresponde a una coordenada la cuál ya ha sido explorada y no forma parte de la solución final
- `public boolean isPARED(int row, int col)`  
Método que determina si la coordenada obtenida para formar el camino solución corresponde a una pared o bloque no transitable
- `public void setVisited(int row, int col, boolean value)`  
Método que indica que la coordenada descrita ya está visitada
- `public boolean isValidLocation(int row, int col)`  
Método que describe si la localización actual es válida usando una fila y columna de la matriz
- `public int[][] printPath(List<Coordenada> path)`  
Método que luego de obtener la lista de coordenadas para el camino solución correspondiente va a generarlo en otra matriz, lo único aquí que se realiza es la impresión del camino dentro de la matriz de 0 y 1
- `public int[][] toBinary(int[][] maze)`  
La estructura obtenida anteriormente se la va a transformar en una matriz apta para ser dibujada, pues la anterior para los diferentes parámetros como pared usa números para distinguirlos, de esta forma la solución el laberinto se la puede manejar de mejor forma al trabajar con 0 y 1

### 3.3.20 Métodos MazeGenerator

- `public MazeGenerator(int dim)`  
Método constructor de la clase que obtiene la dimensión del laberinto a crear.
- `public int[][] generateMaze()`

Método que devolverá el laberinto creado y que será en la mayoría de las ocasiones único para resolverlo, sin importar el tamaño de este

- `private boolean validNextNode(Node node)`

Método que determina si el siguiente nodo va a formar parte del camino solución o si es que ya se ha realizado un camino y una pared, sirve principalmente para impedir que los caminos crezcan sin control o que las paredes lo hagan

- `private ArrayList<Node> findNeighbors(Node node)`

Dependiendo de sus vecinos en coordenadas, determinará si se puede añadir al camino que se está generando en el laberinto actual y si es aceptable devolverá una lista de nodos que posterior se usará para seguir agregando al laberinto que se está creando.

- `private Boolean pointOnGrid(int x, int y)`

Determina si la coordenada que se está generado corresponde a una coordenada que a su lado hay un camino o una pared, si hay una pared el valor para la coordenada no será válido y optará por poner a una pieza de camino, por otro lado, si es un bloque optará, por lo contrario.

- `private Boolean pointNotCorner(Node node, int x, int y)`

La única coordenada que para este caso puede corresponder a una posición en una esquina será la solución, por tanto, todos los caminos están dentro de la matriz y no se podrá viajar por sus contornos.

### 3.3.21 Métodos Node

- `Node(int x, int y)`

Método constructor de la clase nodo, que acepta la coordenada x, y

### 3.3.22 Métodos Sonido

- `public Sonido()`

Método constructor de la clase sonido que permite inicializar la ruta en donde se encuentra dicho sonido y posterior lo va a reproducir

- `public void apagar()`

Método que mantiene la posibilidad de apagar la música en un momento determinado

### 3.3.23 Métodos Laberinto

- `public Laberitno(int tamanoLaberinto)`

Método constructor que se usará para inicializar la dimensión del laberinto y algunos parámetros estáticos como lo serán el tamaño del mismo y el laberinto diferente que va a generar.

- `public void pintarLaberinto(Graphics g)`

Método que se lo va a sobrescribir para hacer ser uso de graphics y permitir dibujar tanto al personaje como al laberinto que funcionan en simultaneo con los eventos de movimiento.

### 3.3.24 Métodos Personaje

- `public void pintarPersonaje(Graphics g)`

Mismo uso que la clase laberinto, pero esta vez se lo usa para poder pintar al personaje, la única restricción que se implementa aquí es que los valores retornen a la entrada de los

laberintos que se crean posteriormente debido a que si se lo deja con la última posición entonces el personaje siempre se pinta en el final.

### 3.3.25 Métodos Database

- DataBase(String nameFile)  
Constructor de la clase Database que inicializara el nombre del fichero a guardar los registros de los usuarios agregados
- createFile(String nameFile)  
Crea el fichero binario para agregar a los nuevos usuarios
- final void writeFile()  
Escribe en el fichero binario luego de una operación de registro de usuario
- final void readFile() throws FileNotFoundException, IOException, ClassNotFoundException  
Lee los registros contenidos en el archivo binario para poder ser utilizados
- abstract void tDefault()  
Crea un Usuario por defecto cada vez que el archivo binario se borra por accidente, este usuario es: admin password: 1234

### 3.3.26 Métodos Usuario

- Usuario(String nombreUsuario, String cotrasena)  
método constructor de la clase usuario
- String getNombreUsuario()  
Obtiene el atributo nombre para dicho objeto
- String getCotrasena()  
Obtiene la contraseña de dicho objeto

### 3.3.27 Métodos UsuarioTrs

- public UsuarioTrs()  
Método constructor de la clase UsuarioTrs
- String crearUsuario(Usuario registro) throws MyException  
Crea y envia un nuevo usuario al fichero binario ,también busca si ese usuario existe para no agregarlo nuevamente
- void tDefault()  
Metodo que define su funcionalidad en esta clase, el objetivo es agregar al usuario por defecto
- buscarUsuario(String usuario)  
Busca a un Usuario especifico por nombre de usuario
- List<Usuario> read()  
Lee todos los registros del fichero binario para obtener los usuarios para registros de sesion

### 3.3.28 Métodos Log

- public Log()  
Método constructor de la clase Log
- jButton3ActionPerformed(java.awt.event.ActionEvent evt)  
Metodo que devuelve al usuario encontrado para que pueda acceder
- void jButton4ActionPerformed(java.awt.event.ActionEvent evt)  
Metodo que envia al usuario a una pantalla para registrarse
- void jButton1ActionPerformed(java.awt.event.ActionEvent evt)

### 3.4 Información de Eventos

#### 3.4.1 Eventos JuegoGanado

- `private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)`  
Método que permite abrir la ventana de inicio una vez se haya completado el objetivo del laberinto y que a decidido iniciar una nueva partida
- `private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)`  
Método que permite abrir otra ventana una vez se haya completado el objetivo del laberinto y a decidió salir del programa

#### 3.4.2 Eventos JuegoPerdido

- `private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)`  
Método que permite abrir otra ventana una vez se haya completado el objetivo del laberinto y a decidió salir del programa

#### 3.4.3 Eventos Practicar

- `private void jPanel1KeyPressed(java.awt.event.KeyEvent evt)`  
Comprueba si el Usuario está realizando de manera correcta el tutorial y si es que se esta llegando a la solución de manera satisfactoria, para completar todo el tutorial debe de hacer uso de las flechas y llevarlas hasta el borde de la ventana

#### 3.4.4 Eventos Terminar

- `public void actionPerformed(ActionEvent e)`  
Termina la ejecución de la ventana terminar es decir hace el llamado a dispose, una vez usada la tecla t

#### 3.4.5 Eventos Ventana1

- `private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que valida que los datos que se ingresen no correspondan a letras o en su defecto envíe un mensaje indicando que los valores sobrepasan los valores soportados para la dimensión del laberinto que están entre 3 y 140
- `private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que realiza la correspondiente salida del sistema
- `private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que crea la ventana de práctica si es que se desea realizar nuevamente el tutorial
- `private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que genera la ventana y ayuda a observar algunas reglas y créditos.
- `private void nombreKeyPressed(java.awt.event.KeyEvent evt)`  
Evita que se ingresen caracteres dentro del input, pues este solo permite valores numéricos
- `private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que permite reiniciar la aplicación desde el inicio de la ventana de bienvenida
- `private void nombreKeyPressed(java.awt.event.KeyEvent evt)`  
Evento que envía un mensaje de alerta al ingresar datos inválidos

### 3.4.6 Eventos Abajo

- `private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que elimina cualquier tipo de evento relacionado, es decir cierra la ventana

### 3.4.7 Eventos Arriba

- `private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que elimina cualquier tipo de evento relacionado, es decir cierra la ventana

### 3.4.8 Eventos Derecha

- `private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que elimina cualquier tipo de evento relacionado, es decir cierra la ventana

### 3.4.9 Eventos Izquierda

- `private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que elimina cualquier tipo de evento relacionado, es decir cierra la ventana

### 3.4.10 Eventos Inicio

- `private void progresoStateChanged(javax.swing.event.ChangeEvent evt)`  
Método que permitirá realizar los procesos de desvanecimiento en las ventanas de bienvenida durante un periodo de tiempo, el cual luego cambiará a la siguiente ventana a presentar.

### 3.4.11 Eventos Fin

- `private void progresoStateChanged(javax.swing.event.ChangeEvent evt)`  
Evento que sirve para controlar el tiempo que se ejecutará la ventana final del programa

### 3.4.12 Eventos FinPractica

- `private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)`  
Evento que sirve para controlar el tiempo que se ejecutará la ventana final del programa

## 4 Librerías programadas

### 4.1 Librería Forms

Comentario: Objetos padre y reutilizable

Archivo: Librería JFrame

Archivo: Librería jl 1.0.1













### 4.1.1 Lista de Objetos

Aplicación

LaberintoERA



Lista de Forms

-  Inicio
-  JuegoGanado
-  JuegoPerdido
-  LaberintoGUI
-  Practicar
-  InformacionInicio
-  Valoresvalidos
-  FinPractica
-  Log
-  Registrar

**VentanaSplash**



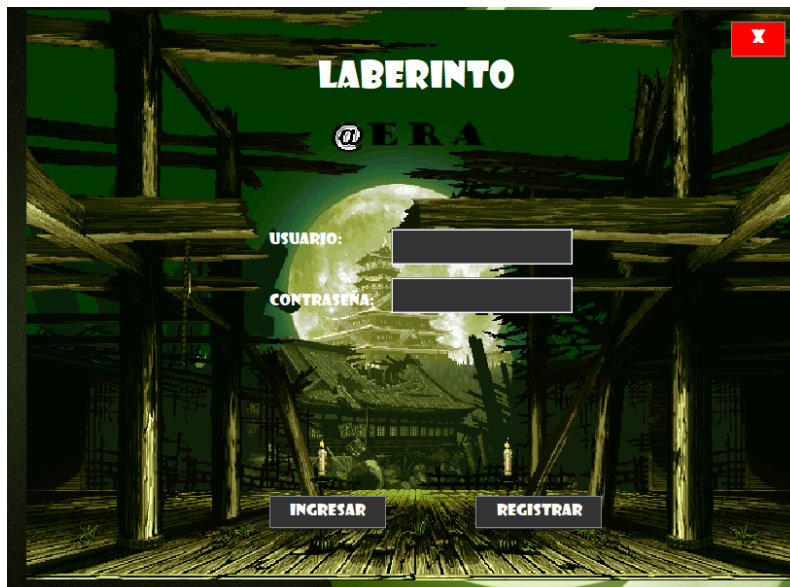
**Comentario:** Splash Screen que se usa para dar la bienvenida al cliente se hace uso de Thread.sleep con un intervalo de 25ms que se repite 100 veces mientras un gif se va moviendo en la pantalla, mientras tanto se carga el componente para realizar el tutorial.

Atributos

- ✓ Fondo
- ✓ JLabel

Componente	Característica
Jlabel1	BackGround [240,240,240]  font      SHOWCARD GOTHIC 36 BOLD  preferredSize    [420, 406]  icon      laberinto.jpg

### VentanaLog



**Comentario:** Ventana usada para que el usuario ingrese tanto al juego o como para que pueda registrarse y posterior entrar con su usuario y contraseña

### Atributos

- JButton
- JLabel
- JTextField

Componente	Características
jTextField	background [51,51,51]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [14, 25]
jLabel1	background [255,255,255]  font SHOWCARD GOTHIC 36 BOLD  preferredSize 17  text Usuario
jLabel2	background [255,255,255]  font SHOWCARD GOTHIC 36 BOLD  preferredSize 17  text Contraseña
jLabel2	background [255,255,255]  font SHOWCARD GOTHIC 36 BOLD  preferredSize 19  text Laberinto @ERA
jButton1	background [51,51,51]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [93, 33]

	text Ingresar
jButton2	background [51,51,51]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [93, 33]  text Registrar
jButton3	background [255,0,0]  font Tahoma 11 Plain  preferredSize [42, 35]  text X

### Ventana Registrar



**Comentario:** Ventana usada para que el nuevo usuario pueda registrarse en el sistema y que posterior pueda acceder a una partida dentro del juego

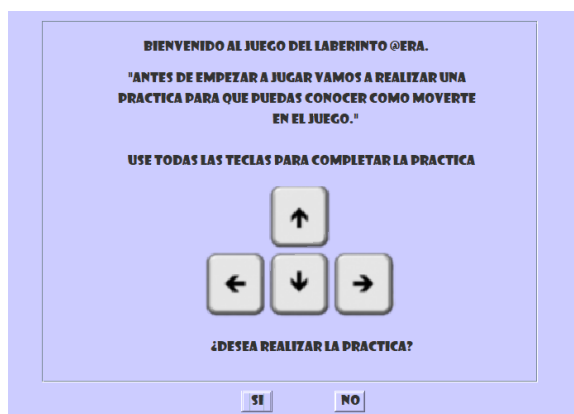
### Atributos

- JButton
- JLabel
- JTextField

Componente	Características
jTextField	background [51,51,51]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [14, 25]
jLabel1	background [255,255,255]  font SHOWCARD GOTHIC 36 BOLD  preferredSize 17  text Nombre de Usuario
jLabel2	background [255,255,255]  font SHOWCARD GOTHIC 36 BOLD  preferredSize 17  text Contraseña
jLabel3	background [255,255,255]  font SHOWCARD GOTHIC 36 BOLD  preferredSize 17  text Confirmar Contraseña

jLabel3	<p>background [255,255,255]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>preferredSize 19</p> <p>text Registrate en el Laberinto @ERA</p>
jButton1	<p>background [51,51,51]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>preferredSize [93, 33]</p> <p>text Registrar</p>
jButton2	<p>background [51,51,51]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>preferredSize [93, 33]</p> <p>text Regresar</p>

## Ventana Práctica



**Comentario:** Ventana usada para indicar el inicio del tutorial si es que así lo desea, esta parte se la puede omitir

#### Atributos

- JButton
- JLabel
- JTextArea

Componente	Características
jTextArea	background [204,204,240]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [540, 407]  icon checked_user_opt.png
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  preferredSize 13  text USE TODAS LAS TECLAS PARA COMPLETAR LA PRACTICA
jLabel2	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  preferredSize 13  text jLabel1
jLabel2	background [63,63,95]

	<p>font SHOWCARD GOTHIC 36 BOLD</p> <p>preferredSize 13</p> <p>text Bienvenido al Juego del Laberinto @ERA</p>
jLabel2	<p>background [63,63,95]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>text "Antes de empezar a jugar vamos a realizar una</p>
jButton2	<p>background [63,63,95]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>preferredSize [22,21]</p> <p>text Si</p>
jButton1	<p>background [240,240,240]</p> <p>font Tahoma 11 Plain</p> <p>preferredSize [22,21]</p> <p>text No</p>

## Registro





**Comentario:** Ventana usada para realizar el correspondiente tutorial para moverse dentro del laberinto

#### Atributos

- ✓ JTextField
- ✓ JLabel

Componente	Características
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text Completa la practica
jLabel2	background [240,240,240]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [72, 22]

jLabel3	background [240,240,240]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [960, 640]
---------	--

## Interfaz



## Atributos

- ✓ JButton
- ✓ JTextArea
- ✓ JLabel
- ✓ JPanel

Comentario: Ventana que indica que cada una de las fases del tutorial se están realizando correctamente

Componente	Caracteristicas
jButton1	background [240,240,240]

	preferredSize [61, 21]  font SHOWCARD GOTHIC 36 BOLD  text No
JTextArea	background [204,204,255]  font Segoe UI 12 Plain  preferredSize [252, 174]  text -
jButton1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [61, 21]  text ACEPTAR
jPanel1	background [255,255,255]  foreground [0,0,0]  border null  preferredSize [178, 477]
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text Bienvenido a la practica del juego
jLabel3	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text "Con esta tecla podras moverte hacia aRRIBA."

jLabel2	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text –
---------	---

## Instrucciones



**Comentario:** Ventana donde se da una ligera explicación de cual es la meta del jugador en el juego

## Atributos

- ✓ JButton
- ✓ JLabel
- ✓ JTextArea

Componentes	Características
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text "Felicidades ha completado la practica Disfruta

jLabel2	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text La aventura del Laberinto, Podras digitar el nivel
jLabel3	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text de dificultad del laberinto"
jLabel4	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text Para ver la solucion del juego presiona la tecla
jLabel5	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text "Y"
JTextArea	background [204,204,255]  font Segoe UI 12 Plain  preferredSize [252, 174]  text -
jButton1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [61, 21]  text ACEPTAR

## Pantalla principal



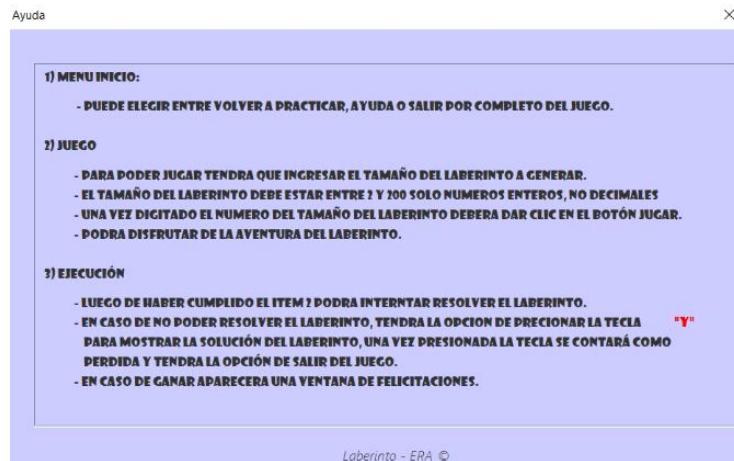
**Comentario:** Ventana donde se indica el tamaño del laberinto que se genera luego de empezar con el botón jugar

### Atributos

- ✓ JButton
- ✓ JLabel
- ✓ JTextArea
- ✓ JPanel
- ✓ JMenuBar
- ✓ JTextField

Componentes	Características
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text Bienvenido al Juego del Laberinto
jLabel2	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text Digita el tamaño del laberito:
jLabel3	background [63,63,95]

	font SHOWCARD GOTHIC 36 BOLD  text de dificultad del laberinto"
JPanel	background [204,204,255]  font Segoe UI 12 Plain  preferredSize [252, 174]  text -
jButton1	background [204,204,255]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [79, 31]  text ACEPTAR
jTextField	background [153,153,153]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [64, 35]



**Comentario:** Ventana donde se da una referencia de la totalidad de la aplicación, se accede a través de la barra de menú en el inicio del programa

### Atributos

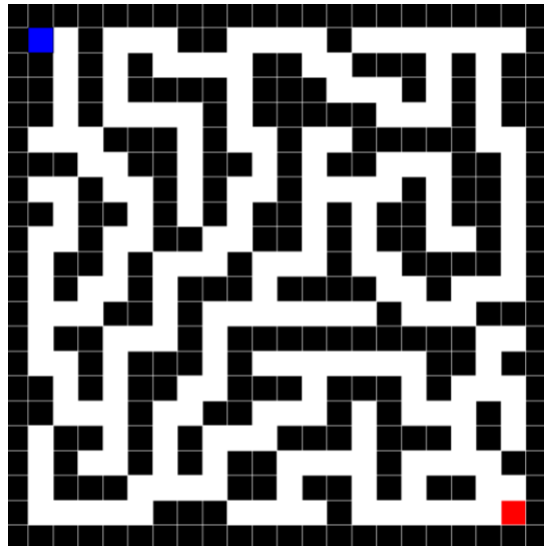
- ✓ JLabel
- ✓ JTextArea

Componentes	Caracteristicas
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text 1) Menu Inicio:
jLabel2	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text - Puede elegir entre volver a Practicar, Ayuda o Salir por completo del Juego.
jLabel3	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text 2) Juego
jLabel4	background [63,63,95]



	<p>font SHOWCARD GOTHIC 36 BOLD</p> <p>text - Para poder jugar tendra que ingresar el tamaño del laberinto a generar.</p>
jLabel5	<p>background [63,63,95]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>text - El tamaño del laberinto debe estar entre 2 y 200 solo numeros enteros, NO decimales</p>
jLabel6	<p>background [63,63,95]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>text - Una vez digitado el numero del tamaño del laberinto debera dar clic en el botón Jugar.</p>
jLabel7	<p>background [63,63,95]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>text - Podra disfrutar de la aventura del Laberinto.</p>
jLabel8	<p>background [63,63,95]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>text 3) Ejecución</p>
jLabel9	<p>background [63,63,95]</p> <p>font SHOWCARD GOTHIC 36 BOLD</p> <p>text 3) Ejecución</p>
jLabel10	<p>background [63,63,95]</p>

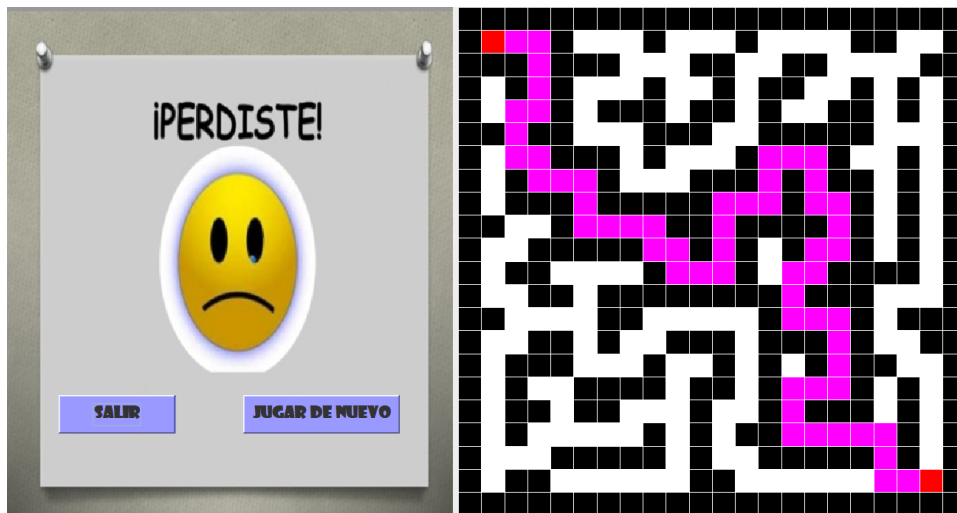
	font      SHOWCARD GOTHIC 36 BOLD  text - Luego de haber cumplido el item 2 podra interntar resolver el laberinto.
jLabel11	background      [63,63,95]  font      SHOWCARD GOTHIC 36 BOLD  text - En caso de no poder resolver el Laberinto, tendra la opcion de precionar la tecla
jLabel12	background      [63,63,95]  font      SHOWCARD GOTHIC 36 BOLD  text - para mostrar la solución del Laberinto, una vez presionada la tecla se contará como
jLabel13	background      [63,63,95]  font      SHOWCARD GOTHIC 36 BOLD  text perdida y tendra la opción de salir del Juego.
JTextArea	background      [204,204,255]  font                  Segoe UI 12 Plain  preferredSize      [252, 174]  text      -



**Comentario:** Esta ventana representa el laberinto que se genera una vez quiera iniciar el juego, los componentes de esta ventana no se los usa directamente de la paleta, sino, son pintados con el método sobrescrito `graph` que proporciona el frame.

#### Atributos

Componentes	Características
JFrame	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  preferer size [753, 443]



**Comentario:** Ventana donde se indica que el usuario a presionado la tecla para observar el camino y por tanto a perdido el juego

#### Atributos

- ✓ JButton
- ✓ JLabel
- ✓ JTextArea
- ✓ JPanel

Componentes	Características
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text Perdiste
JPanel	background [204,204,255]  font Segoe UI 12 Plain  preferredSize [252, 174]  text -
jButton1	background [204,204,255]  font SHOWCARD GOTHIC 36 BOLD

	preferredSize [79, 31]  text SALIR
jTextArea	background [153,153,153]  font SHOWCARD GOTHIC 36 BOLD  preferredSize [64, 35]



**Comentario:** Ventana donde se indica que el usuario ha llegado al final del laberinto y por tanto lo ha ganado.

#### Atributos

- ✓ JButton
- ✓ JLabel
- ✓ JTextArea
- ✓ JPanel

Componentes	Características
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text Perdiste

JPanel	background [204,204,255] font Segoe UI 12 Plain preferredSize [252, 174] text -
jButton1	background [204,204,255] font SHOWCARD GOTHIC 36 BOLD preferredSize [79, 31] text SALIR
jTextArea	background [153,153,153] font SHOWCARD GOTHIC 36 BOLD preferredSize [64, 35]



**Comentario:** Ventana que indica que ha sido el fin del juego, se presenta únicamente cuando el usuario quiere salir o a perdido la partida

#### Atributos

- ✓ JLabel
- ✓ JTextArea
- ✓ JPanel

Componentes	Características
jLabel1	background [63,63,95]  font SHOWCARD GOTHIC 36 BOLD  text LABERINTO @ERA
JPanel	background [204,204,255]  font Segoe UI 12 Plain  preferredSize [300, 194]  text -

## 5 Anexos

