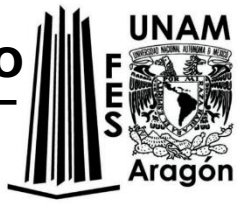
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO



FACULTAD DE ESTUDIOS SUPERIORES ARAGON

LABERINTO

P R E S E N T A

Alexis Hernández Zamudio

APROFESOR

Jesús Hernández Cabrera

Gpo:1158

URL del repositorio:

https://github.com/TyrBalder1439/Estructura-de-Datos-

Ciudad Nezahualcóyotl, EDOMEX. 11 de OCTUBRE del 2024

Laberinto.java | Pila.java | Array2D.java

```java
public class Array2D {  6 usages  felipe herman
    private int[][] array;  6 usages

    public Array2D(int filas, int columnas) {  1 usage  felipe herman
        array = new int[filas][columnas];
    }

    public int getValor(int fila, int columna) {  1 usage  felipe herman
        return array[fila][columna];
    }

    public void setValor(int fila, int columna, int valor) {  2 usages  felipe herman
        array[fila][columna] = valor;
    }

    public int getFilas() {  1 usage  felipe herman
        return array.length;
    }

    public int getColumnas() {  1 usage  felipe herman
        return array[0].length;
    }

    public void mostrarArray() {  1 usage  felipe herman
        for (int[] fila : array) {
            for (int celda : fila) {
                System.out.print(celda + " ");
            }
            System.out.println();
        }
    }
}
```

Laberinto.java | Pila.java | Array2D.java

```java
import java.util.ArrayList;

public class Pila<T> {  2 usages  felipe herman
    private ArrayList<T> elementos;  7 usages

    public Pila() {  1 usage  felipe herman
        elementos = new ArrayList<>();
    }

    public void push(T elemento) {  5 usages  felipe herman
        elementos.add(elemento);
    }

    public T pop() {  1 usage  felipe herman
        if (!estaVacia()) {
            return elementos.remove(elementos.size() - 1);
        }
        return null;
    }

    public T peek() {  1 usage  felipe herman
        if (!estaVacia()) {
            return elementos.get(elementos.size() - 1);
        }
        return null;
    }

    public boolean estaVacia() {  3 usages  felipe herman
        return elementos.isEmpty();
    }
}
```

Laberinto.java × Pila.java Array2D.java

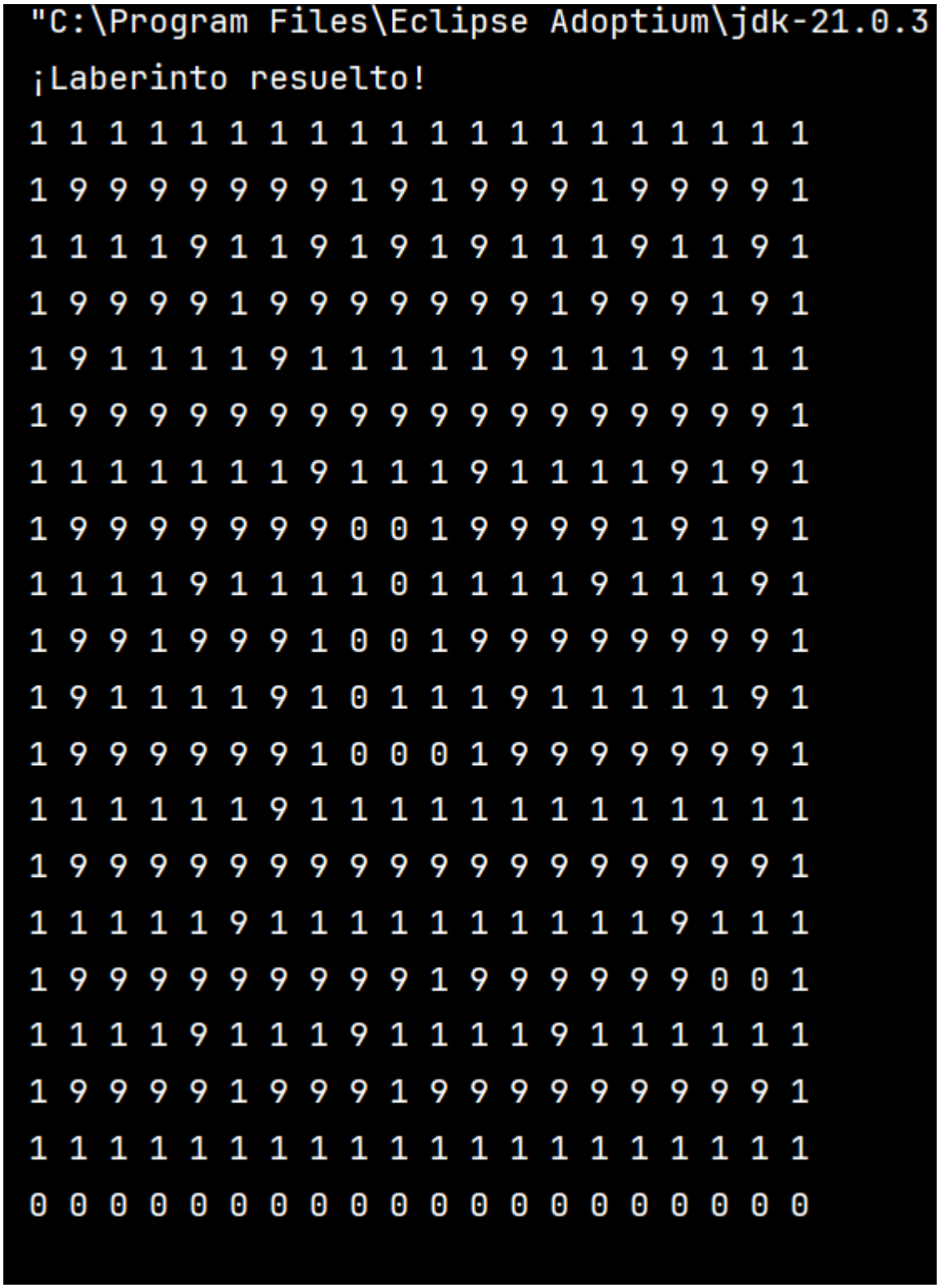
```java
public class Laberinto {  felipe herman
    private Array2D laberinto;  6 usages
    private int[] entrada;  2 usages
    private int[] salida;  3 usages
    private Pila<int[]> pila;  9 usages

    public Laberinto(Array2D laberinto, int[] entrada, int[] salida) {  1 usage
        this.laberinto = laberinto;
        this.entrada = entrada;
        this.salida = salida;
        this.pila = new Pila<>();
    }

    public boolean resolverLaberinto() {  1 usage  felipe herman
        pila.push(entrada);
        while (!pila.estaVacia()) {
            int[] posicionActual = pila.peek();
            int fila = posicionActual[0];
            int columna = posicionActual[1];

            laberinto.setValor(fila, columna, 9);

            if (fila == salida[0] && columna == salida[1]) {
                return true;
            }

            if (mover(fila, columna - 1)) {
```

```
            if (mover(fila, columna - 1)) {
                pila.push(new int[]{fila, columna - 1});
            } else if (mover(fila - 1, columna)) {
                pila.push(new int[]{fila - 1, columna});
            } else if (mover(fila, columna + 1)) {
                pila.push(new int[]{fila, columna + 1});
            } else if (mover(fila + 1, columna)) {
                pila.push(new int[]{fila + 1, columna});
            } else {
                pila.pop();
            }
        }
        return false;
    }

    private boolean mover(int fila, int columna) {  4 usages  felipe herman
        return (fila >= 0 && fila < laberinto.getFilas() && columna >= 0 && columna < laberinto.getColumnas() && laberinto
    }

    public void mostrarLaberinto() {  1 usage  felipe herman
        laberinto.mostrarArray();
    }

    public static Array2D configurarLaberinto(int filas, int columnas) {  1 usage  felipe herman
```

```
    public static Array2D configurarLaberinto(int filas, int columnas) {  1 usage  felipe herman
        if (filas < 20 || columnas < 20) {
            throw new IllegalArgumentException("El laberinto debe ser al menos de 20x20.");
        }

        Array2D laberinto = new Array2D(filas, columnas);

        int[][] layout = {
                {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
                {1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1},
                {1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1},
                {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1},
                {1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1},
                {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
                {1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1},
                {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1},
                {1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1},
                {1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1},
                {1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1},
                {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1},
                {1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
                {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
                {1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1},
                {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1},
                {1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1},
                {1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
```

```
            {1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1},
            {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
        };

        for (int i = 0; i < layout.length; i++) {
            for (int j = 0; j < layout[i].length; j++) {
                laberinto.setValor(i, j, layout[i][j]);
            }
        }

        return laberinto;
    }
```

```
    public static void main(String[] args) {  felipe herman
        Array2D laberinto = configurarLaberinto(20, 20);
        int[] entrada = {1, 1};
        int[] salida = {17, 18};

        Laberinto lab = new Laberinto(laberinto, entrada, salida);
        if (lab.resolverLaberinto()) {
            System.out.println("¡Laberinto resuelto!");
        } else {
            System.out.println("No se encontró una solución.");
        }

        lab.mostrarLaberinto();
    }
}
```


"C:\Program Files\Eclipse Adoptium\jdk-21.0.3
¡Laberinto resuelto!
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 9 9 9 9 9 9 9 1 9 1 9 9 9 1 9 9 9 9 1
1 1 1 1 9 1 1 9 1 9 1 9 1 1 1 9 1 1 9 1
1 9 9 9 9 1 9 9 9 9 9 9 9 1 9 9 9 1 9 1
1 9 1 1 1 1 9 1 1 1 1 1 9 1 1 1 9 1 1 1
1 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 1
1 1 1 1 1 1 1 9 1 1 1 9 1 1 1 1 9 1 9 1
1 9 9 9 9 9 9 9 9 0 0 1 9 9 9 9 1 9 1 9 1
1 1 1 1 9 1 1 1 1 0 1 1 1 1 9 1 1 1 9 1
1 9 9 1 9 9 9 1 0 0 1 9 9 9 9 9 9 9 9 1
1 9 1 1 1 1 9 1 0 1 1 1 9 1 1 1 1 1 9 1
1 9 9 9 9 9 9 1 0 0 0 1 9 9 9 9 9 9 9 1
1 1 1 1 1 1 9 1 1 1 1 1 1 1 1 1 1 1 1 1
1 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 1
1 1 1 1 1 9 1 1 1 1 1 1 1 1 1 9 1 1 1
1 9 9 9 9 9 9 9 9 9 9 1 9 9 9 9 9 9 0 0 1
1 1 1 1 9 1 1 1 9 1 1 1 1 9 1 1 1 1 1 1
1 9 9 9 9 1 9 9 9 1 9 9 9 9 9 9 9 9 9 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0