Captura del codigo:

```java
package ico.fes.aragon.unam.mx.clases;

import java.util.LinkedList;

public class Pila<T> {  4 usages  wwe-rgh
    private LinkedList<T> stack;  7 usages

    public Pila() {  1 usage  wwe-rgh
        stack = new LinkedList<>();
    }


    public boolean estVacio() {  1 usage  wwe-rgh
        boolean res = false;
        if(this.stack.size()== 0){
            res = true;

        }
        return res;
}
    public int longitud(){  no usages  wwe-rgh
        return this.stack.size();
    }
    public void push(T item) {  2 usages  wwe-rgh
        stack.addFirst(item);
```

```java
public int longitud(){  no usages  wwe-rgh
    return this.stack.size();
}
public void push(T item) {  2 usages  wwe-rgh
    stack.addFirst(item);
}
public char pop() {  2 usages  wwe-rgh
    stack.removeLast();
    return 0;
}
public T peek() {  no usages  wwe-rgh
    return stack.getFirst();
}

@Override  wwe-rgh
public String toString() {
    return "Pila{" +
            "stack=" + stack +
            '}';
}
```

```java
public class Array2d {  5 usages  new *
    private int[][] data;  5 usages
    private int filas;  1 usage
    private int columnas;  1 usage
    public Array2d(int filas, int columnas) {  2 usages  new *
        this.data = new int[filas][columnas];
    }

    public int getData(int x, int y) {  2 usages  new *
        return data [x][y];
    }

    public void setData(int x, int y, int valor) {  28 usages  new *
        this.data[x][y] = valor;
    }

    public int getFilas() {  1 usage  new *
        return data.length;
    }

    public void setFilas(int filas) {  no usages  new *
        this.filas = filas;
    }
```

```java
    public void setFilas(int filas) {  no usages  new *
        this.filas = filas;
    }

    public int getColumnas() {  no usages  new *
        return data[0].length;
    }

    public void setColumnas(int columnas) {  no usages  new *
        this.columnas = columnas;
    }

}
```

```java
import java.util.ArrayList;
import java.util.List;

public class Backtracking {  3 usages  wwe-rgh
    public List<List<Integer>> generador(int[] nums) {  1 usage  wwe-rgh
        List<List<Integer>> resultado = new ArrayList<>();
        backtracking(resultado, new ArrayList<>(), nums);
        return resultado;
    }

    private void backtracking(List<List<Integer>> resultado, List<Integer> temporal, int[] nums) {  2 usages  wwe-rgh
        // Si la lista temporal tiene el mismo tamaño que nums, se ha encontrado una permutación
        if (temporal.size() == nums.length) {
            resultado.add(new ArrayList<>(temporal));
        } else {
            for (int i = 0; i < nums.length; i++) {
                if (temporal.contains(nums[i])) continue; // Evita duplicados
                temporal.add(nums[i]); // Añade el número a la permutación
                backtracking(resultado, temporal, nums); // Recurre con el número añadido
                temporal.remove( index: temporal.size() - 1); // Retrocede, eliminando el último número añadido
            }
        }
    }
```

```java
public class Laberinto {  wwe-rgh *
    private static Array2d laberinto = new Array2d( filas: 5, columnas: 5);  27 usages
    private static final int N = laberinto.getFilas();  4 usages
    private static Array2d solucion  = new Array2d(N,N);  4 usages
    private static final int[] movFila = {0,-1,0,1};  1 usage
    private static final int[] movColumna = {-1,0,1,0};  1 usage


    public static void main(String[] args) {  wwe-rgh *
        configurarLaberinto();
        int inicioX = 0, inicioY = 0;
        int finX= 4, finY = 4;
        Pila<String> Laberintopilas = new Pila<>();
        if(resolverlaberinto( inicioX, inicioY, finX, finY, Laberintopilas)){
            System.out.println("Camino encontrado");
            imprimirSolucion();
            System.out.println("\n recorrido completo");
            while (!Laberintopilas.estVacio()) {
                System.out.println(Laberintopilas.pop());

            }
        }else{
                System.out.println("Camino no encontrado");
            }
    }
```

```java
private static void configurarLaberinto() {  1 usage  new *
    laberinto.setData( x: 0, y: 0, valor: 0);
    laberinto.setData( x: 0, y: 1, valor: 1);
    laberinto.setData( x: 0, y: 2, valor: 0);
    laberinto.setData( x: 0, y: 3, valor: 0);
    laberinto.setData( x: 0, y: 4, valor: 0);
    laberinto.setData( x: 1, y: 0, valor: 0);
    laberinto.setData( x: 1, y: 1, valor: 1);
    laberinto.setData( x: 1, y: 2, valor: 0);
    laberinto.setData( x: 1, y: 3, valor: 1);
    laberinto.setData( x: 1, y: 4, valor: 0);
    laberinto.setData( x: 2, y: 0, valor: 0);
    laberinto.setData( x: 2, y: 1, valor: 0);
    laberinto.setData( x: 2, y: 2, valor: 0);
    laberinto.setData( x: 2, y: 3, valor: 1);
    laberinto.setData( x: 2, y: 4, valor: 0);
    laberinto.setData( x: 3, y: 0, valor: 0);
    laberinto.setData( x: 3, y: 1, valor: 1);
    laberinto.setData( x: 3, y: 2, valor: 1);
    laberinto.setData( x: 3, y: 3, valor: 1);
    laberinto.setData( x: 3, y: 4, valor: 0);
    laberinto.setData( x: 4, y: 0, valor: 0);
    laberinto.setData( x: 4, y: 1, valor: 0);
    laberinto.setData( x: 4, y: 2, valor: 0);
```

```java
    laberinto.setData( x: 4, y: 0, valor: 0);
    laberinto.setData( x: 4, y: 1, valor: 0);
    laberinto.setData( x: 4, y: 2, valor: 0);
    laberinto.setData( x: 4, y: 3, valor: 0);
    laberinto.setData( x: 4, y: 4, valor: 0);
}

public static boolean resolverlaberinto(int x, int y,int finx,int finy, Pila<String> Laberintopila){  2 usages  wwe-rgh *
        if(x == finx && y == finy){
            solucion.setData(x,y, valor: 1);

            Laberintopila.push( item: "{"+x+","+y+")");
            return true;
        }
        if(esValido(x,y)){
            solucion.setData(x,y, valor: 1);// Marcar como parte del camino
            Laberintopila.push( item: "{" + x + ", " + y + ")");  // Agregar la posición a la pila
        for (int i =0; i<4; i++){
            int nuevoX = x+movFila[i];
            int nuevoY = y+movColumna[i];

            if(resolverlaberinto(nuevoX, nuevoY, finx,finy,Laberintopila)){
                return true;
```

```
            return true;
        }

    }
    solucion.setData(x,y, valor: 1);
    Laberintopila.pop();

}

//}
    return false;


private static boolean esValido(int x, int y) {  1 usage   wwe-rgh *
    return (x >= 0 && x < N && y >= 0 && y < N && laberinto.getData(x,y)==0 && solucion.getData(x,y)== 0);

}

private static void imprimirSolucion() {  1 usage   wwe-rgh *
    Backtracking backtracking = new Backtracking();
    int[] nums = {0,1,0,0,1};
    List<List<Integer>> LaberintoPilas = backtracking.generador(nums);
```

```
        int[] nums = {0,1,0,0,1};
        List<List<Integer>> LaberintoPilas = backtracking.generador(nums);

        // Imprime todas las permutaciones generadas
        for (List<Integer> lista : LaberintoPilas) {
            System.out.println(lista);//
        }
    }

}
```

```
Camino encontrado

 recorrido completo
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Process finished with exit code 0
```