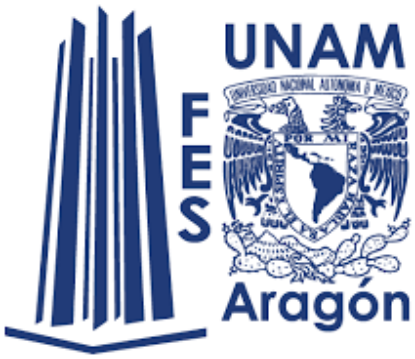


Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Aragón

Ingeniería en Computación

Asignatura: Estructura de datos

Punto extra: Laberinto

Profesor: Jesús Hernández Cabrera

Alumno: Juan Diego Ortiz Cruz

Grupo: 1360

Fecha: 20/09/2024

Capturas Clase pilaADT:

```typescript
export class PilaADT<E> {
    private datos: ListaLigada<E>;

    constructor() {
        this.datos = new ListaLigada<E>();
    }

    public estaVacia(): boolean {
        return this.datos.getLongitud() === 0;
    }

    public longitud(): number {
        return this.datos.getLongitud();
    }

    public peek(): E | null {
        return this.datos.getPrimero();
    }

    public push(elemento: E): void {
        this.datos.agregarAlInicio(elemento);
    }

    public pop(): E | null {
        return this.datos.eliminarElPrimero();
    }

    public toString(): string {
        if (this.estaVacia()) {
            return "";
        }

        let result = "";
        let current = this.datos.getPrimero();
        let index = 0;

        while (current !== null) {
            result += current?.toString();
            current = this.datos.obtener(index + 1);
            index++;
        }
        return result;
    }
}
```

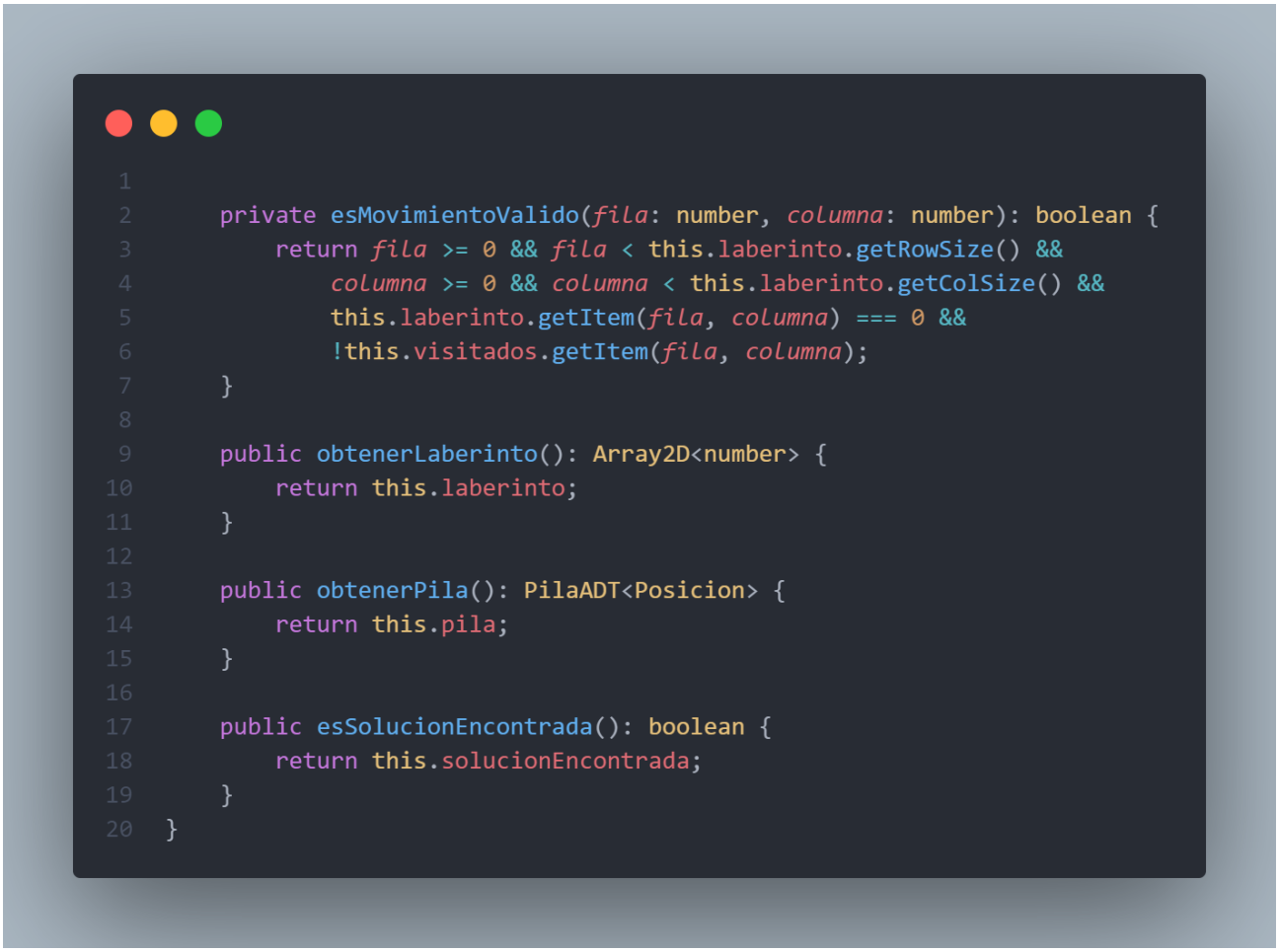
Captura clase LaberintoSolver:

```ts
import { Array2D } from './Array2D.js';
import { PilaADT } from './pilaADT.js';

interface Posicion {
    fila: number;
    columna: number;
}

export class LaberintoSolver {
    private laberinto: Array2D<number>;
    private visitados: Array2D<boolean>;
    private entrada: Posicion;
    private salida: Posicion;
    private pila: PilaADT<Posicion>;
    private solucionEncontrada: boolean;

    constructor(filas: number, columnas: number) {
        this.laberinto = new Array2D<number>(filas, columnas, 1);
        this.visitados = new Array2D<boolean>(filas, columnas, false);
        this.entrada = { fila: 0, columna: 0 };
        this.salida = { fila: filas - 1, columna: columnas - 1 };
        this.pila = new PilaADT<Posicion>();
        this.solucionEncontrada = false;
    }

    public configurarLaberinto(laberintoConfig: number[][]): void {
        if (laberintoConfig.length !== this.laberinto.getRowSize() ||
            laberintoConfig[0].length !== this.laberinto.getColSize()) {
            throw new Error("La configuración del laberinto no coincide con el tamaño inicializado");
        }

        for (let i = 0; i < laberintoConfig.length; i++) {
            for (let j = 0; j < laberintoConfig[i].length; j++) {
                this.laberinto.setItem(i, j, laberintoConfig[i][j]);
            }
        }
    }

    public iniciarResolucion(): void {
        this.pila.push(this.entrada);
        this.visitados.setItem(this.entrada.fila, this.entrada.columna, true);
        this.laberinto.setItem(this.entrada.fila, this.entrada.columna, 2);
    }

```

Continuación de la clase LaberintoSolver:

```
public siguientePaso(): boolean {
    if (this.solucionEncontrada || this.pila.estaVacia()) {
        return false;
    }

    const posicionActual = this.pila.peek()!;

    if (this.esSalida(posicionActual)) {
        this.solucionEncontrada = true;
        return false;
    }

    const siguientePosicion = this.encontrarSiguienteMovimiento(posicionActual);

    if (siguientePosicion) {
        this.pila.push(siguientePosicion);
        this.visitados.setItem(siguientePosicion.fila, siguientePosicion.columna, true);
        this.laberinto.setItem(siguientePosicion.fila, siguientePosicion.columna, 2);
    } else {
        const posicionDescartada = this.pila.pop()!;
        this.laberinto.setItem(posicionDescartada.fila, posicionDescartada.columna, 3);
    }

    return true;
}

private esSalida(posicion: Posicion): boolean {
    return posicion.fila === this.salida.fila && posicion.columna === this.salida.columna;
}

private encontrarSiguienteMovimiento(posicion: Posicion): Posicion | null {
    const movimientos = [
        { fil: 0, col: -1 }, // Izquierda
        { fil: -1, col: 0 }, // Arriba
        { fil: 0, col: 1 },  // Derecha
        { fil: 1, col: 0 }   // Abajo
    ];

    for (const movimiento of movimientos) {
        const nuevaFila = posicion.fila + movimiento.fil;
        const nuevaColumna = posicion.columna + movimiento.col;

        if (this.esMovimientoValido(nuevaFila, nuevaColumna)) {
            return { fila: nuevaFila, columna: nuevaColumna };
        }
    }

    return null;
}
```

Terminación de la clase LaberintoSolver:



```
    private esMovimientoValido(fila: number, columna: number): boolean {
        return fila >= 0 && fila < this.laberinto.getRowSize() &&
            columna >= 0 && columna < this.laberinto.getColSize() &&
            this.laberinto.getItem(fila, columna) === 0 &&
            !this.visitados.getItem(fila, columna);
    }

    public obtenerLaberinto(): Array2D<number> {
        return this.laberinto;
    }

    public obtenerPila(): PilaADT<Posicion> {
        return this.pila;
    }

    public esSolucionEncontrada(): boolean {
        return this.solucionEncontrada;
    }
}
```

Capuras de la implemntacion de la simulacion en JavaScript:

```javascript
let solver, intervalId;
let velocidad = 200;
let pasos = 0;

const laberintoElement = document.getElementById('laberinto');
const pilaElement = document.getElementById('pila');
const iniciarButton = document.getElementById('iniciar');
const pasoButton = document.getElementById('paso');
const reiniciarButton = document.getElementById('reiniciar');
const velocidadSlider = document.getElementById('velocidad');
const velocidadValor = document.getElementById('velocidad-valor');
const pasosElement = document.getElementById('pasos');
const estadoElement = document.getElementById('estado');

function inicializarLaberinto() {
    solver = new LaberintoSolver(21, 25);
    solver.configurarLaberinto(laberintoConfig);
    pasos = 0;
    solver.iniciarResolucion();
    actualizarUI();
}

function actualizarUI() {
    const laberinto = solver.obtenerLaberinto();
    const pila = solver.obtenerPila();

    // Actualizar laberinto
    laberintoElement.innerHTML = '';
    laberintoElement.style.gridTemplateColumns = `repeat(${laberinto.getColSize()}, var(--cell-size))`;

    for (let i = 0; i < laberinto.getRowSize(); i++) {
        for (let j = 0; j < laberinto.getColSize(); j++) {
            const celda = document.createElement('div');
            celda.className = 'celda';
            if (i === 0 && j === 0) {
                celda.classList.add('entrada');
            } else if (i === laberinto.getRowSize() - 1 && j === laberinto.getColSize() - 1) {
                celda.classList.add('salida');
            } else {
                switch (laberinto.getItem(i, j)) {
                    case 0: celda.classList.add('libre'); break;
                    case 1: celda.classList.add('pared'); break;
                    case 2: celda.classList.add('actual'); break;
                    case 3: celda.classList.add('descartado'); break;
                }
            }
            laberintoElement.appendChild(celda);
        }
    }

```

```
    // Actualizar pila
    pilaElement.innerHTML = '';
    let pilaHTML = '';
    let pilaTemp = new PilaADT();
    while (!pila.estaVacia()) {
        const pos = pila.pop();
        pilaHTML += `<div>(${pos.fila}, ${pos.columna})</div>`;
        pilaTemp.push(pos);
    }
    while (!pilaTemp.estaVacia()) {
        pila.push(pilaTemp.pop());
    }
    pilaElement.innerHTML = pilaHTML;

    // Actualizar estadísticas
    pasosElement.textContent = pasos;
    estadoElement.textContent = solver.esSolucionEncontrada() ? 'Solución encontrada' : 'En progreso';
}

function iniciar() {
    if (!intervalId) {
        intervalId = setInterval(siguientePaso, velocidad);
        iniciarButton.textContent = 'Detener';
        pasoButton.disabled = true;
    } else {
        clearInterval(intervalId);
        intervalId = null;
        iniciarButton.textContent = 'Reanudar';
        pasoButton.disabled = false;
    }
}

function siguientePaso() {
    if (solver.siguientePaso()) {
        pasos++;
        actualizarUI();
    } else {
        clearInterval(intervalId);
        intervalId = null;
        actualizarUI();
        pasoButton.disabled = true;
        if (solver.esSolucionEncontrada()) {
            alert('¡Solución encontrada!');
        } else {
            alert('No se encontró solución');
        }
    }
}

```

```
function reiniciar() {
    if (intervalId) {
        clearInterval(intervalId);
        intervalId = null;
    }
    inicializarLaberinto();
    iniciarButton.textContent = 'Iniciar';
    pasoButton.disabled = false;
}

function actualizarVelocidad() {
    velocidad = parseInt(velocidadSlider.value);
    velocidadValor.textContent = `${velocidad} ms`;
    if (intervalId) {
        clearInterval(intervalId);
        intervalId = setInterval(siguientePaso, velocidad);
    }
}

iniciarButton.addEventListener('click', iniciar);
pasoButton.addEventListener('click', siguientePaso);
reiniciarButton.addEventListener('click', reiniciar);
velocidadSlider.addEventListener('input', actualizarVelocidad);

inicializarLaberinto();
```

Capturas de la simulación en ejecución: