Licenciatura en Ingeniería en Tecnologías de la Información e Innovación Digital

Desarrollo de Software Multiplataforma

Estructura de Datos

UNIDAD II

Estructuras de datos básicas

Especificación de la Pila

Hernández Torrez Alondra Vianney -1224100684

Grupo: GTID 141

Docente:

Gabriel Barrón Rodríguez

Dolores Hidalgo. C.I.N. Gto, Viernes 17 de Octubre de 2025.

Especificación de la Pila

```
 // @author Alondra Vianney Hernandez Torres GTID141
// Define qué operaciones puede hacer la pila sin explicar cómo se implementan
public interface Pila<T> {

    // Agrega un elemento a la pila
    void push(T elemento);

    // Quita y retorna el elemento superior de la pila
    // Lanza excepción si la pila está vacía
    T pop() throws PilaVaciaException;

    // Retorna el elemento superior sin eliminarlo
    // Lanza excepción si la pila está vacía
    T top() throws PilaVaciaException;

    // Retorna true si la pila está vacía
    boolean estaVacia();
}
// Excepción personalizada para manejar errores cuando la pila está vacía
public class PilaVaciaException extends Exception {

    // Constructor que recibe un mensaje de error
    public PilaVaciaException(String mensaje) {
        super(mensaje); // Llamamos al constructor de Exception
    }
}
```

Fase de Implementación (La Clase Concreta) La implementación define cómo se llevan a cabo las operaciones, utilizando una estructura de datos concreta. Esto se hace con una clase que implementa la interfaz.

```
 // @author Alondra Vianney Hernandez Torres GTID141
// Implementación de la pila usando un array
// Aquí definimos cómo se almacenan los elementos y cómo funcionan los métodos
public class PilaArray<T> implements Pila<T> {

    // Array interno para guardar los elementos de la pila
    private T[] elementos;
    // Índice de la cima de la pila
    private int cima;

    // Constructor: inicializa el array y la cima
    @SuppressWarnings("unchecked") // Para evitar advertencias al crear un array genérico
    public PilaArray(int tamaño) {
        elementos = (T[]) new Object[tamaño]; // Creamos el array con el tamaño dado
        cima = 0;                          // Al inicio la pila está vacía
    }

    // Método para verificar si la pila está vacía
    @Override
    public boolean estaVacia() {
        return cima == 0; // Retorna true si no hay elementos
    }

    // Método para agregar un elemento a la pila
    @Override
    public void push(T elemento) {
        if (cima < elementos.length) { // Verifica que haya espacio
```

```
            elementos[cima] = elemento; // Guarda el elemento en la cima
            cima++;                     // Incrementa el índice de la cima
        } else {
            System.out.println("La pila está llena"); // Mensaje si no hay espacio
        }
    }

    // Método para quitar y retornar el elemento superior
    @Override
    public T pop() throws PilaVaciaException {
        if (estaVacia()) { // Si la pila está vacía, lanzamos excepción
            throw new PilaVaciaException("No se puede hacer pop, la pila está vacía");
        }
        cima--; // Bajamos la cima
        T elemento = elementos[cima]; // Guardamos el elemento a retornar
        elementos[cima] = null;       // Limpiamos referencia para el recolector de basura
        return elemento;              // Retornamos el elemento
    }

    // Método para ver el elemento superior sin eliminarlo
    @Override
    public T top() throws PilaVaciaException {
        if (estaVacia()) { // Verificamos que la pila no esté vacía
            throw new PilaVaciaException("No se puede consultar top, la pila está vacía");
        }
        return elementos[cima - 1]; // Retorna el elemento superior
    }
}
```

MAIN

```
 // @author Alondra Vianney Hernandez Torres GTID141
public class Main {
    public static void main(String[] args) {

        // Creamos una pila de enteros (T = Integer) con tamaño 5
        Pila<Integer> pila = new PilaArray<>(5);

        try {
            // Agregamos elementos a la pila
            pila.push(10);
            pila.push(20);
            pila.push(30);

            // Consultamos el elemento superior
            System.out.println("Elemento superior: " + pila.top()); // Debe mostrar 30

            // Quitamos elementos usando pop
            System.out.println("Pop: " + pila.pop()); // 30
            System.out.println("Pop: " + pila.pop()); // 20
            System.out.println("Pop: " + pila.pop()); // 10

            // Intentamos hacer pop en una pila vacía
            System.out.println("Pop: " + pila.pop()); // Esto lanzará excepción

        } catch (PilaVaciaException e) {
            // Capturamos la excepción y mostramos el mensaje
```

```
            System.out.println(e.getMessage());
        }
    }
}
```