Universidad Nacional Autónoma

de México

Facultad de Estudios Superiores

Aragón

Tarea 9: Evaluación de Balanceo de Paréntesis

Estructura de Datos

Prof. Hernández Cabrera Jesús

Grupo: 1306

Angeles Mermejo Octavio Emiliano

Código Fuente Pila:

```java
public class Pila<T> {

    8 usages
    private Nodo<T> tope;
    4 usages
    private int tamano;

    4 usages
    private static class Nodo<T> {
        3 usages
        private T dato;
        2 usages
        private Nodo<T> siguiente;

        1 usage
        public Nodo(T dato) {
            this.dato = dato;
        }
    }

    1 usage
    public Pila() {
        tope = null;
        tamano = 0;
    }

    4 usages
    public boolean isEmpty() {
        return tope == null;
    }

    no usages
    public int length() {
        return tamano;
    }

    // Método para añadir un elemento a la pila (push)
    1 usage
    public void push(T elemento) {
        Nodo<T> nuevoNodo = new Nodo<>(elemento);
        nuevoNodo.siguiente = tope;
        tope = nuevoNodo;
        tamano++;
    }

    // Método para sacar el elemento en la cima de la pila
    1 usage
    public T pop() {
        if (isEmpty()) {
            throw new IllegalStateException("La pila está vacía");
        }
        T elemento = tope.dato;
        tope = tope.siguiente;
        tamano--;
        return elemento;
    }

    // Método para ver el elemento en la cima de la pila sin eliminarlo (peek)
    no usages
    public T peek() {
        if (isEmpty()) {
            throw new IllegalStateException("La pila está vacía");
        }
        return tope.dato;
    }
}
```

Código Fuente BalanceoParentesisLlaves:

```java
package fes.aragon.clases;

3 usages
public class BalanceoParentesisLlaves {

    2 usages
    public boolean estanBalanceados(String expresion) {
        Pila<Character> pila = new Pila<>();

        // Recorrer cada caracter de la expresión
        for (char c : expresion.toCharArray()) {
            // Si es una llave o paréntesis de apertura, lo apilamos
            if (c == '{' || c == '(') {
                pila.push(c);
            }
            // Si es una llave o paréntesis de cierre
            else if (c == '}' || c == ')') {
                // Si la pila está vacía, no hay apertura correspondiente
                if (pila.isEmpty()) {
                    return false;
                }
                // Verificamos si el último elemento en la pila es el correspondiente
                char ultimo = pila.pop();
                if (c == '}' && ultimo != '{' || c == ')' && ultimo != '(') {
                    return false;
                }
            }
        }
        // si la pila está vacía todo esta balanceado
        return pila.isEmpty();
    }
}

```

Código Fuente Main:

```java
package fes.aragon.inicio;

import fes.aragon.clases.BalanceoParentesisLlaves;

public class Main {

    public static void main(String[] args) {
        // Crear una instancia de la clase BalanceoParentesisLlaves
        BalanceoParentesisLlaves verificador = new BalanceoParentesisLlaves();

        // Texto de prueba
        String expresion1 = "{(a + b) * (c + d)}"; // Balanceado
        String expresion2 = "{(a + b) * (c + d)";  // No balanceado

        System.out.println("Expresión 1: " + (verificador.estanBalanceados(expresion1) ? "Balanceado" : "No B
        System.out.println("Expresión 2: " + (verificador.estanBalanceados(expresion2) ? "Balanceado" : "No B
    }
}
```

Ejecución del Programa:

```
Expresión 1: Balanceado
Expresión 2: No Balanceado

Process finished with exit code 0
```