TUDS-A-24 - Estructura de datos y algoritmos - C2
TP N° 3.2 – TAD Pilas

Integrantes	**DNI**
Martín Piriz	34.877.112
Agustín Jofre	38.751.595
Adam Rigg	95.285.064

```java
/**

@author Adam Rigg, Martin Piriz, Agustin Jofre
*/

/*
Unidos no mezclados
Se tienen 2 pilas con elementos ordenados de mayor a menor, se desea
pasar todos los elementos a una tercera pila en la que los elementos sigan
manteniendo el orden de mayor a menor.
*/



public Pila unirPilasOrdenadas(Pila pila1, Pila pila2) {

        Pila pilaFinal = new Pila(pila1.size() + pila2.size());

        while (!pila1.pilaVacia() && !pila2.pilaVacia()) {
            if (pila1.verTope() > pila2.verTope()) {
                pilaFinal.push(pila1.pop());
            } else {
                pilaFinal.push(pila2.pop());
            }
        }

        while (!pila1.pilaVacia()) {
            pilaFinal.push(pila1.pop());
        }

        while (!pila2.pilaVacia()) {
            pilaFinal.push(pila2.pop());
        }

        return pilaFinal;
}
```

```java
/**

@author Adam Rigg, Martin Piriz, Agustin Jofre
*/

/*
Verificación de equilibrio de paréntesis
Diseña un programa que tome una cadena de texto que puede contener
varios tipos de paréntesis, incluyendo (), {}, [], y <> y determine si los
paréntesis están equilibrados en la cadena. Es decir, cada paréntesis de
apertura tiene su par correspondiente de cierre y están correctamente
anidados. Utiliza una pila para realizar la verificación. Por ejemplo, la
cadena "{[(a+b)*c]-(d+e)}" debería ser considerada como  equilibrada,
mientras que "[{()}]" también lo es, pero "[(])" no lo es.
 */
package eda.tp3_2;

import java.util.Objects;

/**
 *
 * @author Adam, Martin, Agustin
 */
public class PilaParentesis {
    private Boolean esEquilibrada;
    private Character[] arrParentesis;
    private static char[][] parentesis = {{'(',')'},{'{','}'},{'[',']'},{'<','>'}};
    private enum P{
        OPEN,
        CLOSE
    }
    private Pila pila;

    public PilaParentesis(String cadena) {
        this.arrParentesis = getParentesis(cadena);
        pila = new Pila(arrParentesis.length);

        if(cadena == null || "".equals(cadena) || arrParentesis.length < 1){
            esEquilibrada = null;
```

```
    }
    if(arrParentesis.length % 2 == 1){
        esEquilibrada = false;
        return;
    }

    int i=0;
    Character car;

    while(i<arrParentesis.length){
        car = arrParentesis[i];

        if(esParenApertura(car)){
            pila.push(car);
        }

        if(esParenCierre(car)){
            if(pila.pilaVacia()){
                esEquilibrada = false;
                return;
            }
            if(!Objects.equals(
                                getParenApertura(car),
                                  (char)pila.pop()
                              )
              ){
                esEquilibrada = false;
                return;
            }
        }
        i++;
    }
    if(pila.cursor > -1){
        esEquilibrada = false;
        return;
    }
    esEquilibrada = true;
    return;
}
```

```java
public static Character[] getParentesis(String cadena) {
    if (cadena == null || cadena.equals("")) {
        return null;
    }
    int i = 0;
    String strParen = "";
    for (char car : cadena.toCharArray()) {

        for(i=0; i<parentesis.length; i++){
            if (car == parentesis[i][P.OPEN.ordinal()]) {
                strParen += car + "";
            }
            if (car == parentesis[i][P.CLOSE.ordinal()]) {
                strParen += car + "";
            }
        }

    }
    char[] charParen = strParen.toCharArray();
    Character[] carEnvolturaParen = new Character[charParen.length];
    int j = 0;
    for (char car : charParen) {
        carEnvolturaParen[j] = car;
        j++;
    }
    return carEnvolturaParen;
}


public Boolean esEquilibrada() {
    return esEquilibrada;
}

public boolean esParenApertura(Character car){
    for (char[] paren : parentesis) {
        if (car == paren[P.OPEN.ordinal()]) {
            return true;
        }
```

```
    }
    return false;
}

public boolean esParenCierre(Character car){
    for (char[] paren : parentesis) {
        if (car == paren[P.CLOSE.ordinal()]) {
            return true;
        }
    }
    return false;
}

public Character getParenCierre(Character parenApertura){
    int i = 0;
    for(char[] paren: parentesis){
        if(parenApertura == paren[P.OPEN.ordinal()]){
            return parentesis[i][P.CLOSE.ordinal()];
        }
        i++;
    }
    return null;
}

public static Character getParenApertura(Character parenCierre){
    int i = 0;
    for(char[] paren: parentesis){
        if(parenCierre == paren[P.CLOSE.ordinal()]){
            return parentesis[i][P.OPEN.ordinal()];
        }
        i++;
    }
    return null;
}

@Override
public String toString() {
    if(esEquilibrada == null){
        return "es nulo";
```

```
        }
        if(!esEquilibrada){
            return "no es equilibrada.";
        }
        return "es equilibrada.";
    }

}
```