Estructura de Datos 20/21

Práctica 3

El tokenizer

Objetivos

- Utilizar una pila para solucionar el problema del *tokenizer*
- Reforzar la habilidad de aprender a aprender

Descripción del laboratorio

¿Os habéis preguntado alguna vez cómo se lo hacen los IDE (*Integrated Development Environment*) para identificar si nos falta o nos sobra un paréntesis o llave, o si nos equivocamos, por ej. escribimos) cuando era }?

Este problema es un clásico en las Estructuras de Datos. Para solucionarlo, se utiliza una pila. Y al algoritmo que nos permite identificar los "tokens" (símbolos) y comprobar que la expresión formada por los tokens es correcta, se le conoce como tokenizer.

Tareas a realizar

El objetivo es implementar un método que, dado un ArrayList de elementos de tipo Symbol, nos permita identificar si la expresión formada por el vector de Symbol es correcta en tiempo lineal.

1. ( ) Correcto
2. ((Incorrecto
3. ({ }) Correcto
4. ({ (} Incorrecto

¿Por qué son las expresiones (1) y (3) correctas?

En el programa, tendremos las diferentes clases, que representaran Symbol

- ClosingLlave -> }
- ClosingParentesis –>)
- OpenLlave –> {
- OpenParentesis –> (

Estas clases son Symbol, que es una clase abstracta.

Por tanto, las tareas a realizar son

1. Implementar la clase Symbol y las 4 subclases que heredan de ella
2. Implementar el método comprovarExpressio que encontraréis en el fichero **Tokenizer**.java. Este método trabajará con una Stack<T> de la JCF – no haréis la vista vosotros - para devolver cierto si la expresión es correcta, o falso en caso contrario.

```
public static boolean comprovarExpressio(ArrayList<Symbol> arr_simbols)
```

Podéis implementar métodos auxiliares.

Ayuda

El algoritmo pasa por entender el manejo de la pila en esta situación. La pila la podemos utilizar para ir guardando símbolos. El último símbolo estará en el top de la pila. Tenemos símbolos que abren y cierran. Si nos llega un símbolo de tipo cierre que no se corresponde con el símbolo de tipo apertura en el top de la pila…

Entrega

Fecha: ver la tarea correspondiente en el Campus Virtual

Material: un fichero ZIP con el proyecto IntelliJ IDEA y un documento (RTF, DOC o PDF) con la estrategia del algoritmo.

Criterios (generales) de evaluación

- Práctica no entregada o entregada fuera de plazo = 0
- Entrega parcial, <= 3
- Entrega completa, entre 4 y 10. Se valorará el diseño y claridad del código, comentarios, etc.