Ozmar René Escobar Avila - 201602988 24/03/2019
Laboratorio Arqui 1

Manual Técnico

INTRODUCCION

Bienvenido al manual técnico
En este documento encontrará información relevante sobre el tecnicismo sobre esta aplicación

Esta aplicación fue creada para un sistema operativo **Linux**, es el ensamblador de Microsoft, y es capaz de crear aplicaciones compatibles con cualquier sistema operativo desarrollado por Microsoft, en este caso la plataforma objetivo es DOSBox

Se está compilando bajo la versión **OpenJDK 11**.

La aplicación consiste de un IDE, que sirve de interfaz para un analizador léxico que procede a generar una entrada para el analizador sintáctico, que a su vez genera un arbol AST que le sirve al analizador semántico para hacer validaciones mientras ejecuta cada instrucción.

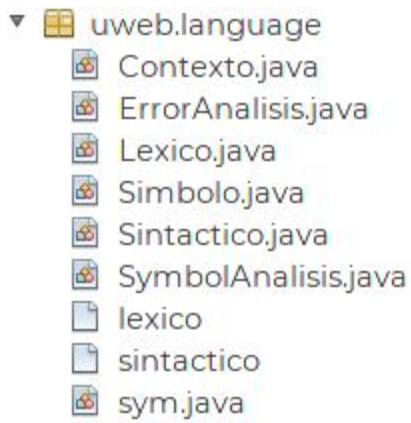
El analizador léxico reconoce 2 lenguajes distintos por medio de estados, cuando se encuentra la etiqueta <?HS se cambia de estado para reconocer el lenguaje procedural, hasta encontrar la etiqueta de cierre ?>

El analizador sintáctico se encarga de generar el AST por medio de listas enlazadas.

El analizador semántico hace uso intensivo del patrón de diseño **visitador**, en donde se tiene una interfaz que se llama **Instruccion.java** que tiene un método llamado **ejecutar**, y se tienen muchas clases, una clase por cada tipo de instrucción, y cada una hace la implementación de esta interfaz dependiendo del tipo de acción que se quiera realizar, entonces el analizador sintáctico solo se encarga de crear nuevas instancias de cada tipo de instrucción según corresponda, en orden de aparición, entonces si el análisis sintáctico es correcto, se procede a ejecutar cada una de las instrucciones del AST, el procedimiento es **recursivo**, y como lo que se quiere es generar una salida en HTML, se retornan valores según la instrucción, se concatenan en la salida y se genera el archivo html.

Aunque durante esta última fase de análisis, puede existir la posibilidad de errores semánticos, el analizador se encarga de notificar al programador en la **consola de salida**, el error encontrado y una breve descripción de lo que ocurrió, aunque ocurran errores semánticos, se intentó minimizar el daño a la salida mediante un sistema modular, que aborta las operaciones atómicas, cercanas al error semántico y continua ejecutando las demás.

Codigo



Paquete ‘uweb’

Contexto.java: Extiende a la clase LinkedList<Simbolo> y funciona como tabla de símbolos que ayuda al análisis semántico a ejecutar código
ErrorAnalisis.java: Clase que representa un error léxico o sintáctico para uso posterior
Lexico.java: Clase generada de JFlex
Simbolo.java: Clase que representa un símbolo de la fase de análisis semántico
Sintactico.java: Clase generada de CUP
SymbolAnalisis.java: Clase que representa un símbolo del análisis léxico para uso posterior
lexico: Archivo .JFlex
sintactico: Archivo .CUP
sym.java: Clase generada de CUP para hacer interfaz con la clase Lexico.java


app
Cliente.java
Main.java
Tool.java
app.tree
Asignacion.java
Atributo.java
Boton.java
CodigoHS.java
Echo.java
If.java
Imagen.java
Insertar.java
Instruccion.java
Operacion.java
Parrafo.java
Repetir.java
Struct.java
StructBoton.java
StructCabecera.java
StructColumna.java
StructColumnaC.java
StructCompi.java
StructCuerpo.java
StructEspacio.java
StructFila.java
StructImagen.java
StructParrafo.java
StructSalto.java
StructTabla.java
StructTextoA.java
StructTextoB.java
StructTitulo.java
Tabla.java
TextoA.java
TextoB.java
TextoLibre.java

Paquete ‘app’

Cliente.java: Contiene el código del ide y los métodos para generar las salidas
Main.java: Clase punto de entrada del programa
Tool.java: Clase que contiene procedimientos útiles

Paquete ‘app.tree’

Asignacion.java: Clase instrucción que se encarga de la escritura de símbolos al Contexto
Atributo.java: Clase instrucción que estructura los posibles atributos de una etiqueta STRUCT
Boton.java: Clase instrucción que al ejecutarse, se valida semánticamente y se retorna a sí misma, haciendo disponible su instancia, para uso en la tabla de símbolos de Contexto.java, en el método **toString**() se genera la traducción HTML con los atributos que en ese momento se tengan
CodihoHS.java: Clase que almacena el AST del lenguaje procedural HScript
Echo.java: Clase instrucción que posee la estructura para la impresión de una expresión que debe reducirse a una cadena de texto, para imprimirla a la consola local.
If.java: Clase instrucción que posee la estructura para la ejecución de un rombo de decisión, genera una salida de ser necesario
Imagen.java: Clase instrucción que al ejecutarse, se valida semánticamente y se retorna a sí misma, haciendo disponible su instancia, para uso en la tabla de símbolos de Contexto.java, en el método **toString**() se genera la traducción HTML con los atributos que en ese momento se tengan
Insertar.java: Clase instrucción que hace la llamada a los métodos **toString**() de las clases HScript para regresar dicha traducción y concatenarla a la salida final.
Instruccion.java: Interfaz que obliga implementar el método ejecutar, es clave para el patrón visitador implementado en este proyecto
Operacion.java: Clase instrucción que se encarga de reducir operaciones aritméticas, lógicas y de cadena, y regresarlas a donde se requiera.
Parrafo.java: Clase instrucción que al ejecutarse, se valida semánticamente y se retorna a sí misma, haciendo disponible su instancia, para uso en la tabla de símbolos de Contexto.java, en el método **toString**() se genera la traducción HTML con los atributos que en ese momento se tengan
Repetir.java: Clase instrucción que posee la estructura para la ejecución de un ciclo FOR, genera una salida de ser necesario
Struct.java: Clase padre a todos los elementos STRUCT, que contiene métodos comunes y añade comportamientos de diseño
StructBoton.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un boton en HTML

StructCabecera.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un header en HTML
StructColumna.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un TD en HTML
StructColumnaC.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un TH en HTML
StructCompi.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de una etiqueta HTML
StructCuerpo.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un body en HTML
StructEspacio.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un div en HTML
StructFila.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un TR en HTML
StructImagen.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de una imagen y sus atributos en HTML
StructParrafo.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de una etiqueta P en HTML
StructSalto.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un salto de línea en HTML
StructTabla.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de una tabla en HTML
StructTextoA.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un h1 en HTML
StructTextoB.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de un h2 en HTML
StructTitulo.java: Hereda de Struct y contiene código para validarse semánticamente y generar la salida de una etiqueta title en HTML
Tabla.java: Clase instrucción que al ejecutarse, se valida semánticamente y se retorna a sí misma, haciendo disponible su instancia, para uso en la tabla de símbolos de Contexto.java, en el método **toString**() se genera la traducción HTML con los atributos que en ese momento se tengan
TextoA.java: Clase instrucción que al ejecutarse, se valida semánticamente y se retorna a sí misma, haciendo disponible su instancia, para uso en la tabla de símbolos de Contexto.java, en el método **toString**() se genera la traducción HTML con los atributos que en ese momento se tengan
TextoB.java: Clase instrucción que al ejecutarse, se valida semánticamente y se retorna a sí misma, haciendo disponible su instancia, para uso en la tabla de símbolos de Contexto.java, en el método **toString**() se genera la traducción HTML con los atributos que en ese momento se tengan
TextoLibre.java: Clase instrucción que al ejecutarse regresa el texto almacenado