INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

COMPILADORES E INTERPRETES

Documentación Analizador Contextual

Integrante:

Andrés Gutiérrez Salas - 201223823

Profesor:

Ignacio Trejos

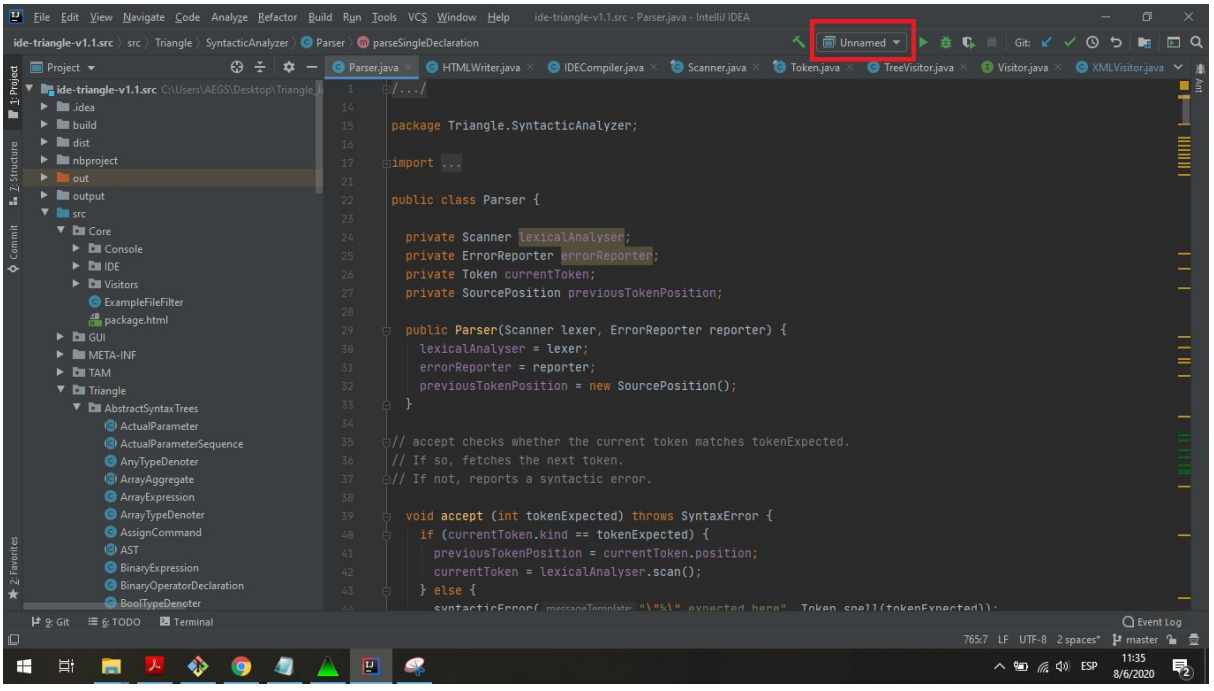
Cartago, Costa Rica **Junio, 08, 2020**

Analizador Contextual

1. Repeat tiene dos componentes que son la expresión y el comando, para la expresión se realiza una visita y se retorna el tipo para verificar que sea del tipo bool. Para el comando se realiza una visita para verificar que todos los componentes interiores están correctamente.
2. Para esto se crea una variable de tipo constante para aprovechar que estas no pueden ser modificadas pero que puede ser leída. Esta variable se inicializa con la primera expresión y se guarda la segunda y para ambas se verifica que ambas sea tipo entero y para terminar se verifica que el comando esté correcto.
3. No se hizo nada para var Id := Exp ya que no se realizaron los extras.
4. Para el Private se crea un nuevo scope en donde se agregan las declaraciones hechas y en IdentificationTable se creó un nuevo closeScopePrivate que se encarga de cerrar el scope para todas las declaraciones privadas que se hayan realizado.
5. Para la recursividad, se crea unos nuevos scopes tanto para abrirlo como para cerrarlo. Se agregó PendingCall y FutureCallExpression qué son las encargadas del manejo de las llamadas recursivas, la primera se encarga del proc y revisar que no queden llamadas pendientes y la segunda se encarga del func que mira a las llamadas de funciones a futuro.
6. No se hizo nada para repeat_loop_in_end ya que no se realizo ningun extra
7. No se hizo nada para exit y next ya que no se realizo ningun extra.
8. No se hizo nada para el return ya que no se realizo ningun extra.
9. Todos los cambios realizados para el private y el rec, tambien se agrego equals para hacer la comparación de objetos en el Identifier.
10. No se detectaron errores contextuales.
11. Se crearon tanto pruebas negativas como positivas para todo lo modificado y agregado y el objetivo de estas es probar que funcionan correctamente y que cuando no lo hacen se despliegue

el mensaje de error correctamente. El resultado es el esperado ya que el programa compila de forma exitosa, se crea el AbstractSyntaxTrees y se pueda ver desde el IDE. También se prueba que el HTML y el XML son generados de forma correcta.

12. No hubo discusión como tal al ser yo el único integrante pero los resultados obtenidos fueron los esperados.
13. Se volvió largo sobretodo para la parte del Private y la recursividad, tanto para proc como para func. Ya que hubo que crear varias cosas que afectan lo demás y por ende tuve que revisar de forma profunda que todo funcionara de forma correcta cada vez que cambie o agregue algo.
14. Soy el único miembro así que yo me encargue de todo.
15. Para compilarlo en Intellij, se tiene que abrir el proyecto, se tiene que ir a Add Configuration que en mi caso ya sale Unnamed porque ya se configuró.



Luego se va a abrir una ventana, ahi se le da al + y se selecciona la opción Application, y ahí luego donde dice Main Class, solo se selecciona la clase Main.java

16. Para la ejecución del programa de forma directa en Windows, solo es necesario correr el **ide-triangle-v1.2.jar** que se encuentra dentro la carpeta **Triangle_Java_IDE_Gutierrez_Andres** ya con eso se abre el IDE de triángulo.

En la misma carpeta **Triangle_Java_IDE_Gutierrez_Andres** esta la carpeta output que es donde se crean los HTML y los XML luego de compilar el programa creado en triángulo.