T2 – Análise Léxica de Compiladores

Descrição: O trabalho 2 (T2) da disciplina consiste em implementar um analisador sintático para a linguagem LA (Linguagem Algorítmica) desenvolvida pelo prof. Jander, no âmbito do DC/UFSCar. O analisador sintático deve ler um programa-fonte e apontar onde existe erro sintático, indicando a linha e o lexema que causou a detecção do erro.

Professor: **Daniel Lucrédio**

Alunos:

- Daniel Watanabe 800697
- Paula Larocca 769705
- Victor Bonometo 800232

Requerimentos:

- Sistema operacional Windows
- Ferramenta IDE APACHE Mavens
- Java JDK

Execução:

1. Inserindo o Antlr4 no Mavens
   É necessário colocar no arquivo **pom.xml**:

   Dependencia do Antlr4 com a versão 4.13.1

```
<dependencies>
    <dependency>
      <groupId>org.antlr</groupId>
      <artifactId>antlr4-runtime</artifactId>
      <version>4.13.1</version>
    </dependency>
  </dependencies>
```

   Plugin do ANTLR4:

```
<plugin>
    <groupId>org.antlr</groupId>
    <artifactId>antlr4-maven-plugin</artifactId>
    <version>4.13.1</version>
    <executions>
     <execution>
      <id>antlr</id>
      <goals>
       <goal>antlr4</goal>
      </goals>
     </execution>
    </executions>
   </plugin>
```

   Plugin do Maven-Jar

```
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.2.0</version>
```

```
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
            <mainClass>grupo.compiladores.t2.T2</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
```

Plugin para adicionar o Manifest:

```
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.2.4</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
          <configuration>
            <transformers>
              <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                <mainClass>grupo.compiladores.t2.T2</mainClass>
              </transformer>
            </transformers>
          </configuration>
        </execution>
      </executions>
    </plugin>
```

E após colocar todos esses plugins e dependências no arquivo **pom.xml**, é necessário colocar a gramática Grama.g4 em um local específico dito pela própria documentação:

/NomeProjeto/src/main/antlr4/pacote/Gramatica.g4

No nosso caso ficou:

/T1/src/main/antlr4/grupo/compiladores/t2/Grama.g4

Depois basta colocar o arquivo T2.java no

/NomeProjeto/src/main/java/pacote/T2.java

No nosso caso ficou:

/T1/src/main/java/grupo/compiladores/t2/T2.java

Além disso, para esse caso foi necessário criar um arquivo SaidaCustom.java, no mesmo diretório do T2.java, ele é quem consegue captar os erros de parser.

E realizar build no projeto, criando o arquivo **T2-1.0-SNAPSHOT.jar**

2. Rodar os casos de testes

Para conseguir rodar o corretor automático deve-se baixar os arquivos necessários dos casos de teste, criar uma pasta temporária para as próprias saídas geradas

java -jar compiladores-corretor-automatico-1.0-SNAPSHOT-jar-with-dependencies.jar "java -jar C:/Users/dahir/OneDrive/Documentos/NetBeansProjects/T2/target/T2-1.0-SNAPSHOT.jar" "C:/MinGW/bin/gcc.exe" "C:/Users/dahir/OneDrive/Documentos/NetBeansProjects/T2/target/temp" "C:/Users/dahir/OneDrive/Documentos/NetBeansProjects/T2/target/casostestes" "ra3, ra2, ra1" "t2"

Gerando a seguinte saída:

```
==================================
Nota do grupo "ra3, ra2, ra1":
CT 1 = 0.0 (0/37)
CT 2 = 10.0 (62/62)
CT 3 = 0.0 (0/9)
CT 4 = 0.0 (0/9)
CT 5 = 0.0 (0/20)
==================================
```