**使用说明**

1.**程序语言及工具**： Java语言，编译器eclipse

2.**文件**： snl.jar为执行文件，后缀.snl为测试文件，测试文件与执行文件放在同级目录可方便打开

3.**输入源程序**： 测试文件可通过菜单左上角 文件->打开 ，也可以直接复制到中间的文本框里

4.**编译**： 点左上角 编译选项->编译 即可完成词法分析、语法分析

5.**结果显示**： 左边：词法分析结果的Token序列

右边：语法分析的结果语法树，如果有语法错误会显示不全

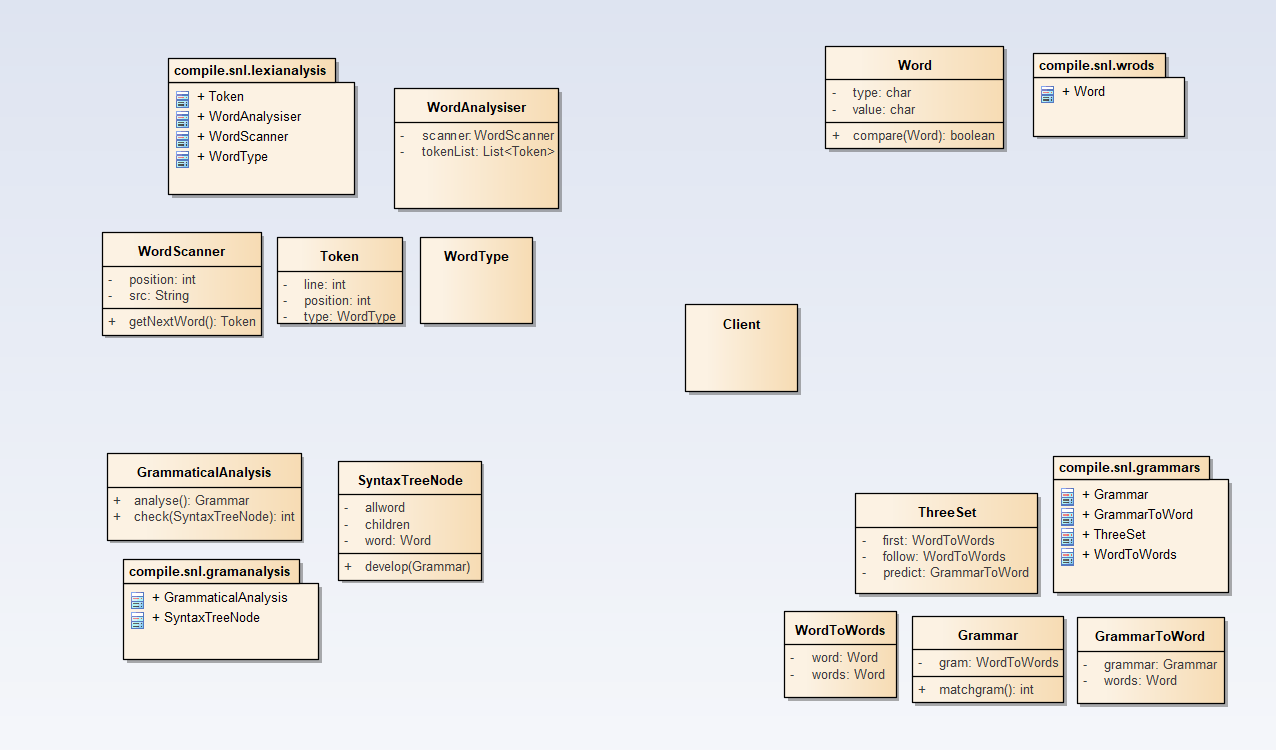
下边：错误信息

**原创性说明：**

本程序完全由小组3人分工独立完成，未粘贴他人代码。

可以证明的地方：求出了first、follow、predict集，语法分析仅有一个函数analyse，即可进行所有语法的分析

**设计思路**



源程序经过WordAnalysiser分析，得到Token序列，由GrammaticalAnalysis根据三个集合ThreeSet及Token序列分析出语法树及错误。

**程序解释：main函数（界面） compile.snl.clinet.Client**

**程序解释\_基础类型介绍**

Word：单词类

WordToWords：一个Word对应多个Word，用于存语法A=>bCd、first集和follow集

Grammar：一个WordToWords对象，存一条语法

GrammarToWord：一个语法对应多个Word，用于存predict集

**程序解释\_词法分析**

compile\snl\lexianalysis包下：

词法分析：

1.**Token**保存3部分信息：单词类型，单词位置（若不为标识符或者常量则为-1），单词在源文件所在行数。

2.**WordAnalysiser**类实际上使用了创建者模式，具体的单词扫描在WordScanner中完成。

3.**WordScanner**类完成单词扫描，主要函数为getNextWord（）

4.**WordType**是一个枚举类，保存了各种单词对应类型。

compile\snl\util包下：

一些工具类

1.**FileUtil**实现把一个文件读入到字符串

2.**WordUtil**实现一些判断：是否为字母，是否为数字，是否为保留字。

compile\snl\clinet包下：

**Client**类负责界面的展示，中间是代码区，左边是token序列展示，右边是语法树展示，下面是错误信息提示

**程序解释\_求3个集合 first/follow/predict**

compile.snl.grammars包下

**ThreeSet**

变量介绍:

1.**first** 存放first集的hashmap,key是非终极符,value是终极符的hashset

2.**follow**存放follow集的hashmap,key是非终极符,value是终极符的hashset

3.**predict**存放predict集的hashmap,key是WordToWords类型的语法产生式,value是终极符的hashset

4.**nonterminal** 存放**终极符**的hashset集合

主要函数介绍:

1. **init**初始化first, follow, predict 集及非终极符

2. **getFirst**遍历每一条产生式，如果右侧第一个是终极符,则添加至左侧终极符的first集， 如果第一个是非终极符,则此元素first集添加至左侧终极符的first集,并判断此first集是否包含空,如果包含则判断下一个符号

3.**getFollow** 遍历每一条产生式,遍历右侧每一个非终极符,把下一个符号的first集加入此非终极符的follow集，最后一个符号如果是终极符,则结束,否则,将左侧非终极符的follow集加入此非终结符的follow集,如果此非终结符的first集包含空,将左侧非终极符的follow集加入前一个非终结符的follow集

4.**getPredict**遍历每一条产生式，如果右侧第一个是终极符,则加入左侧非终极符的predict集，如果是空,则左侧符号的predict集就是它的follow集，否则,循环加空入右侧非终极符的first集,并判断是否包含空,如果全部右侧符号first集包含空,则再将左侧非终极符的follow集加入其predict集;

**程序解释\_语法分析**

compile.snl.gramanalysis包下：

1. **GrammaticalAnalysis** 语法分析类：由同包下的word.txt输入文法，根据Token序列及每个文法的predict集进行语法分析，生成语法树，同时记录错误信息。

主要函数介绍：

1. **analyse** 输入：一个树的节点，该节点使用的语法

根据语法对节点进行递归分析，遍历语法右端进行匹配，如果是ε,跳过；得到符号如果是终极符，如果匹配，将子节点加入父节点的children中；不匹配 抛出异常退出；如果是非终极符A，先根据predict集找到匹配的语法，如果未找到与之匹配的语法，直接退出递归并报出错误信息；否则将子节点加入父节点的children中，再进行对子节点的递归分析。

1. **readToken**读当前token
2. **nextToken**将Token序列指针向后移动一个
3. **match**判断Token和语法的一个终极符是否匹配
4. **foundGrammar**根据predict集找到匹配的语法：判断这个Token是否属于由非终极符A 推导的grammar
5. **init**初始化 并进行语法分析
6. **getTreeMessage** 返回语法树的信息
7. **getErrorMessage** 返回错误信息

2. **SyntaxTreeNode** 树节点：一个word保存当前节点信息，一个ArrayList<SyntaxTreeNode> 类型的children保存子节点的信息

主要函数介绍：**toString**：将树的信息由String返回

compile.snl.util包下

3.**GrammarException**：自定义异常类型，用于语法错误的捕捉及记录。