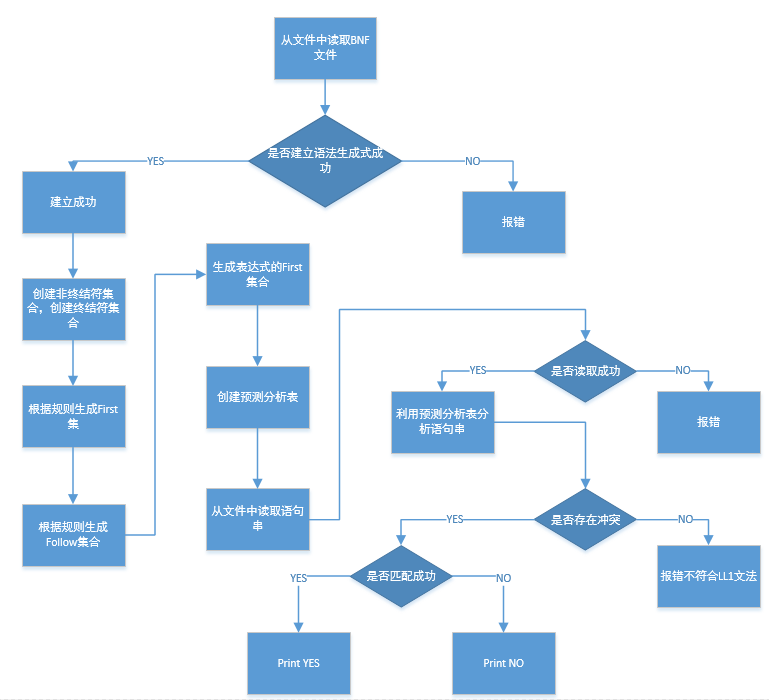
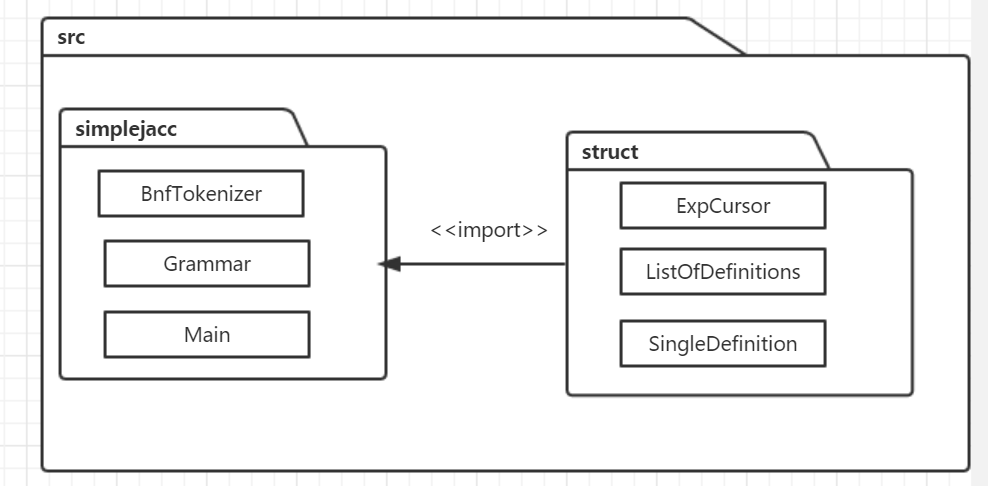
SimpleJacc 文档说明

2014210173 陈世远

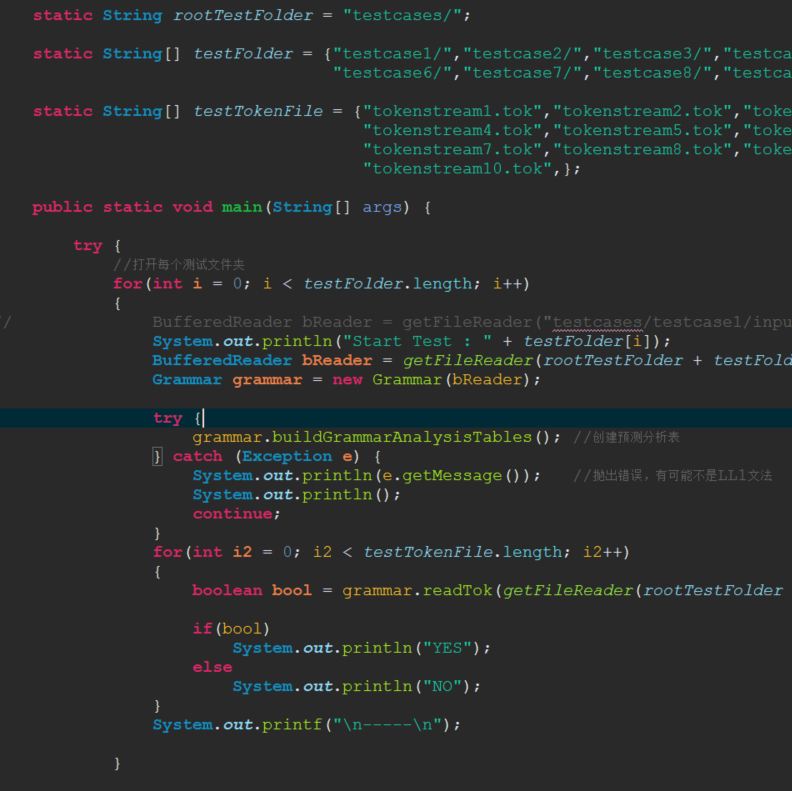
主要实现思路



相关包图、



Main函数（入口函数）



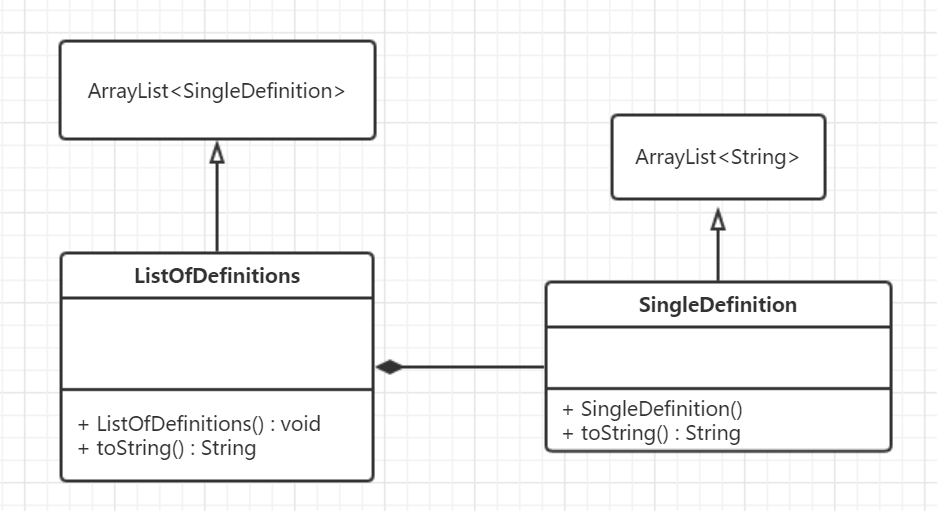
类图（为了方便理解，自行在类图中添加了注释）



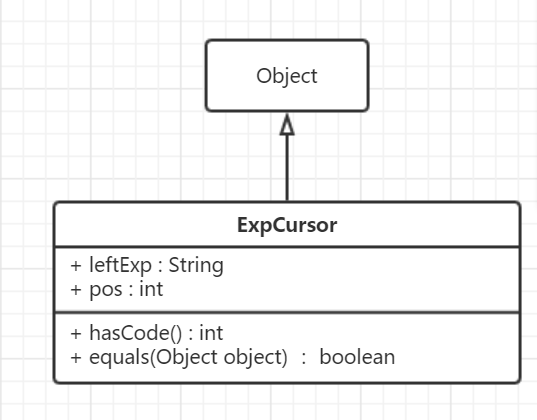
Grammar类本身作为一个语法类，其中包括了所有的生成式，非终结符，终结符，预测分析表的数据成员。重要采用的原始数据结构主要为 Map和ArraryList。其中最为重要的产生式的存储使用的数据结构为**Map<String,ListOfDefinitions>**

并提供了readTok(BufferedReader reader)方法和runGrammar(ArrayList<String> tok)方法来分析输入的语句是否符合这个语法。由于时间的问题，这个类最重要，但是写的并不是很好。

在测试过程中发现一点，如果存在左递归的情况。会崩溃。因为first集和follow集一直递归直到栈溢出。

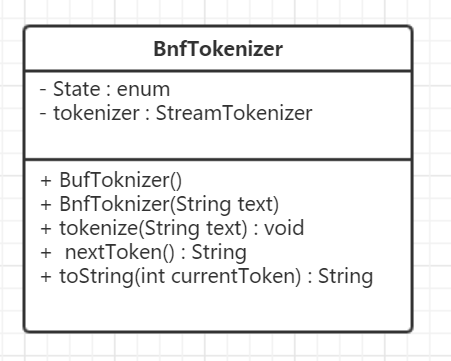


ListOfDefinitions类和SingleDefinition他们均继承于ArrayList。是语法规则产生式的结构元素。



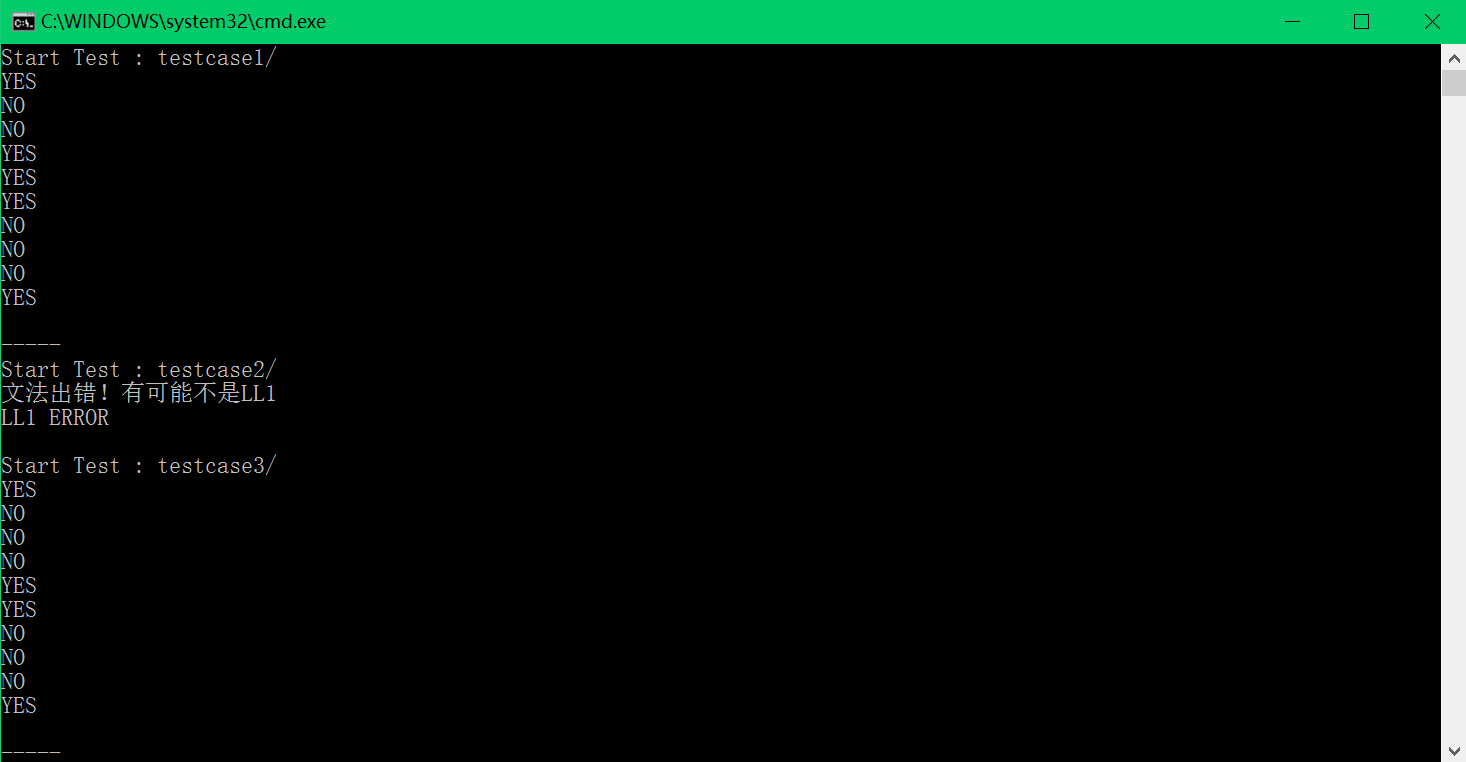
其实写这个类的主要目的是作为在**Map<String,ListOfDefinitions>**中定位产生式的一个结构。

而重写了hasCode()和equals(Object object)是为了可以在Map的数据结构中充当起Key



这个是作为从数据流中解析并读取BNF范式的词法分析类。

## 运行结果



## 总结和思考

本来打算好好的使用面向对象的思想去完成，可是到后期时间各方面的安排原因，导致没有很好的完成。

整个完成的时间，大概有一半的时间在调试的过程中。对编译原理课程的知识如果没有比较熟悉的了解。会花很多的时间在调试。

但是在实践的过程中也加深了对一些知识点的理解。亲手测试，会有不一样的体会。

我想以后有时间的话，说不定会去认真完成一下。因为我在查找资料的时候，发现网络上用Java写的LL1语法分析器的例子很少，十分想写好一点发布到Github上去。