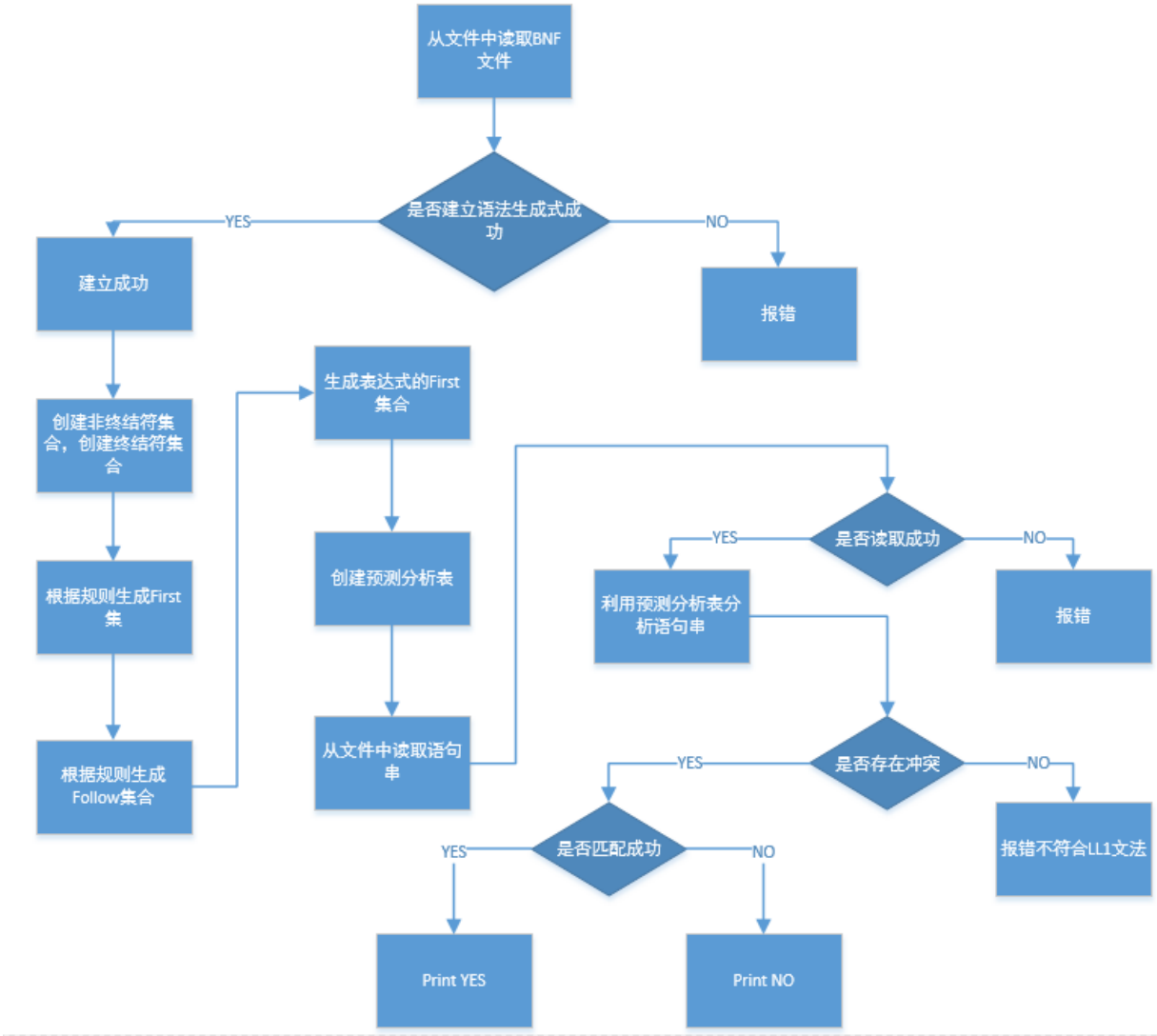
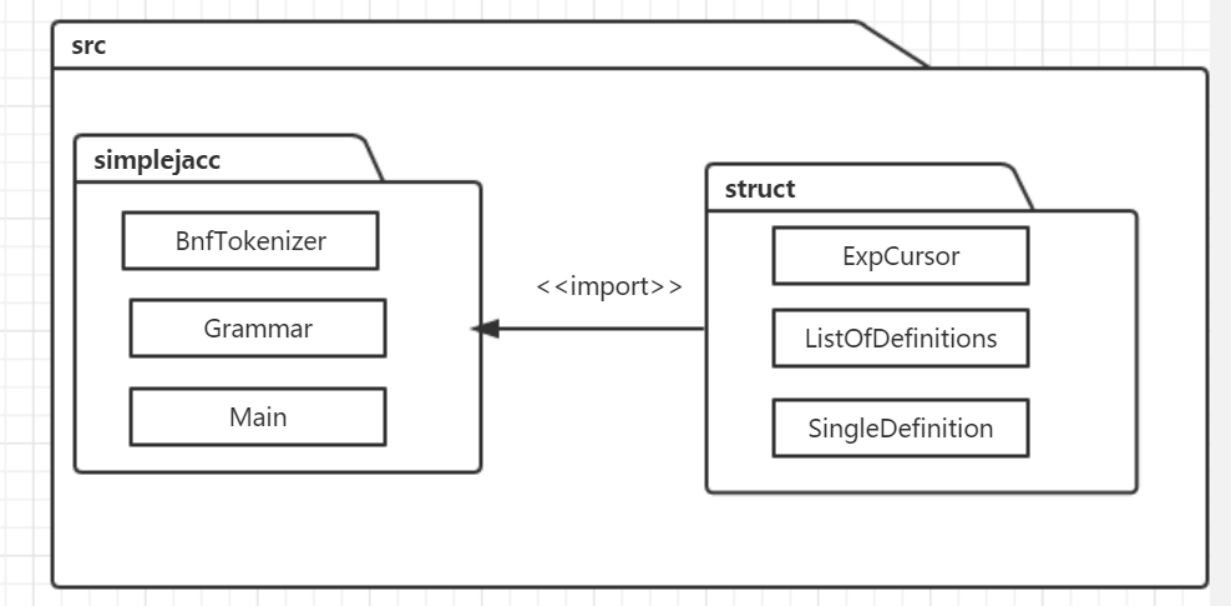
SimpleJacc 文档说明

2014210173 陈世远

主要实现思路



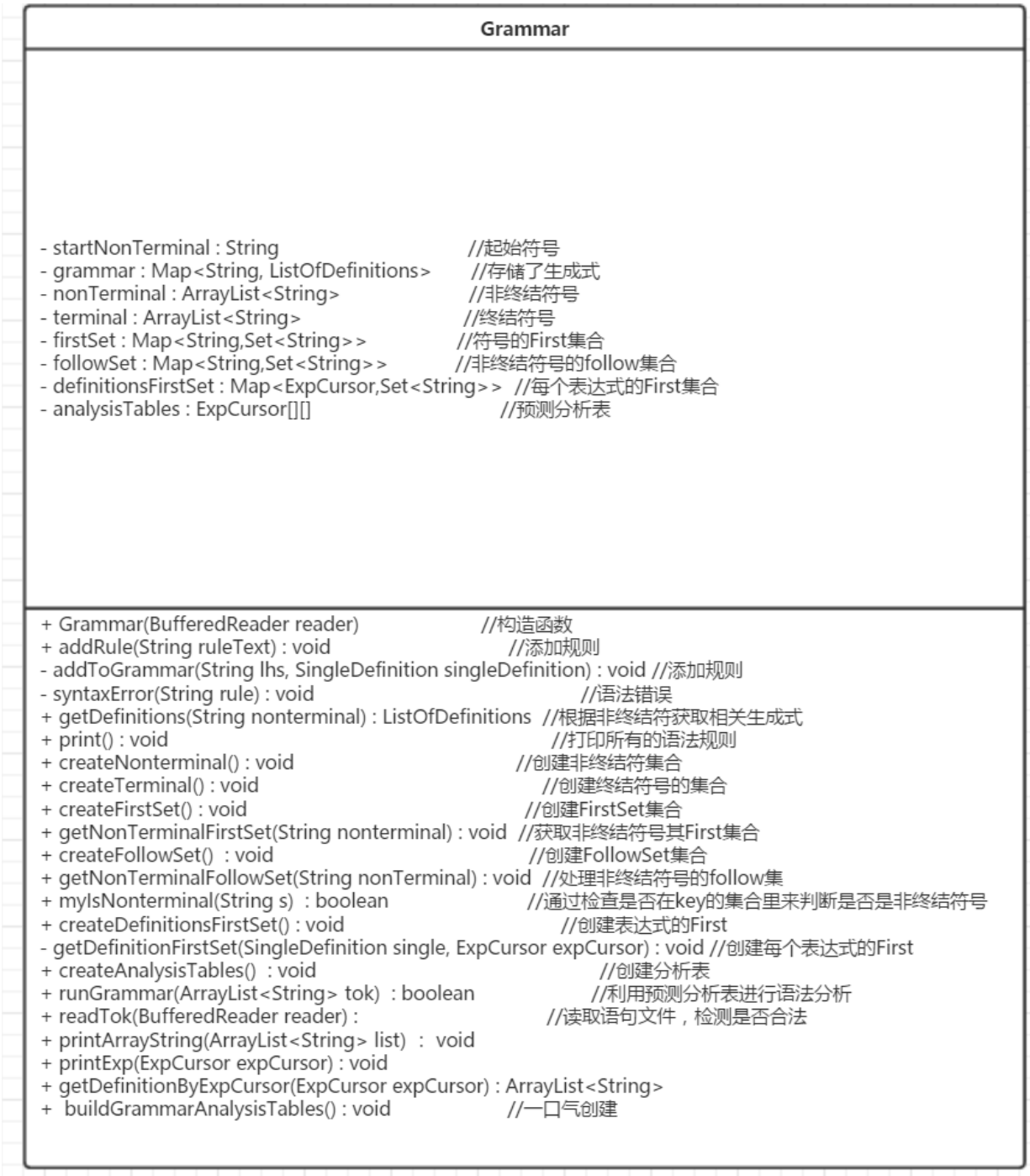
相关包图、



Main 函数（入口函数）

```
static String rootTestFolder = "testcases/";

static String[] testFolder = {"testcase1/","testcase2/","testcase3/","testca
                              "testcase6/","testcase7/","testcase8/","testca

static String[] testTokenFile = {"tokenstream1.tok","tokenstream2.tok","toke
                                 "tokenstream4.tok","tokenstream5.tok","toke
                                 "tokenstream7.tok","tokenstream8.tok","toke
                                 "tokenstream10.tok",};

public static void main(String[] args) {

    try {
        //打开每个测试文件夹
        for(int i = 0; i < testFolder.length; i++)
        {
            BufferedReader bReader = getFileReader("testcases/testcase1/inpu
            System.out.println("Start Test : " + testFolder[i]);
            BufferedReader bReader = getFileReader(rootTestFolder + testFold
            Grammar grammar = new Grammar(bReader);

            try {
                grammar.buildGrammarAnalysisTables(); //创建预测分析表
            } catch (Exception e) {
                System.out.println(e.getMessage());    //抛出错误，有可能不是LL1文法
                System.out.println();
                continue;
            }
            for(int i2 = 0; i2 < testTokenFile.length; i2++)
            {
                boolean bool = grammar.readTok(getFileReader(rootTestFolder

                if(bool)
                    System.out.println("YES");
                else
                    System.out.println("NO");
            }
            System.out.printf("\n-----\n");

        }
```

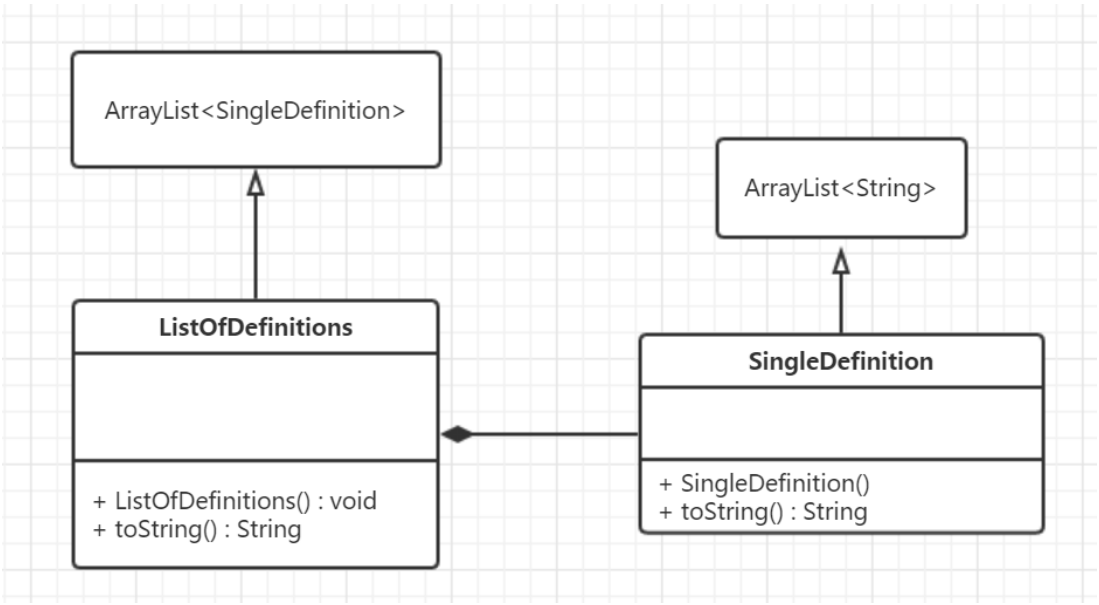
类图（为了方便理解，自行在类图中添加了注释）



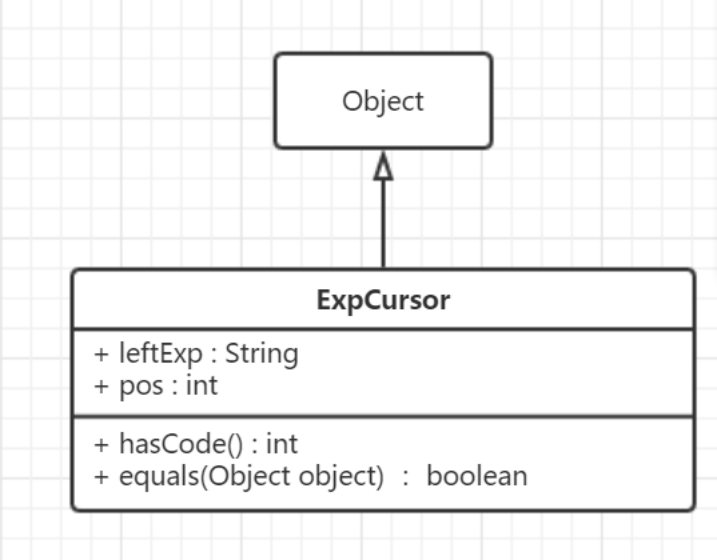
Grammar 类本身作为一个语法类，其中包括了所有的生成式，非终结符，终结符，预测分析表的数据成员。重要采用的原始数据结构主要为 Map 和 ArraryList。其中最为重要的产生式的存储使用的数据结构为 **Map<String,ListOfDefinitions>**

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。

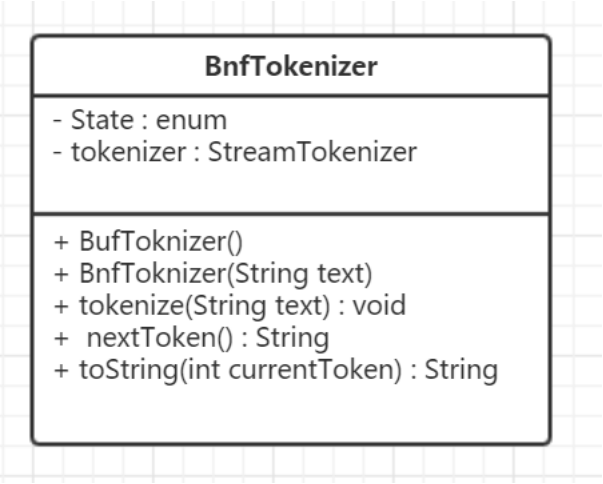


ListOfDefinitions 类和 SingleDefinition 他们均继承于 ArrayList。是语法规则产生式的结构元素。



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。