编译系统课程实验报告

实验 1：词法分析

姓名	王翰坤	院系	计算学部	学号	1183710106
任课教师	单丽莉		指导教师	单丽莉	
实验地点	格物楼 207		实验时间	4 月 17 日	
实验课表现	出勤、表现得分		实验报告得分		实验总分	
	操作结果得分					

一、需求分析

得分

设计实现类高级语言的词法分析器，基本功能如下：

（1）能识别以下几类单词：

a) 标识符（由大小写字母、数字以及下划线组成，但必须以字母或者下划线开头）

b) 关键字（①类型关键字：整型、浮点型、布尔型、记录型；②分支结构中的 `if` 和 `else`；③循环结构中的 `do` 和 `while`；④过程声明和调用中的关键字）

c) 运算符（①算术运算符；②关系运算符；③逻辑运算）

d) 界符（①用于赋值语句的界符，如 `=` ；②用于句子结尾的界符，如 `;` ；③用于数组表示的界符，如 `[` 和 `]`；④用于浮点数表示的界符 `.` ）

e) 常数，包括无符号整数（含八进制和十六进制数）、浮点数（含科学计数法）、字符串常数等。

f) 注释（`/*……*/` 形式）

（2）能够进行简单的错误处理，即识别出测试用例中的非法字符。程序在输出错误提示信息时，需要输出具体的错误类型（即词法错误）、出错的位置（源程序行号）以及相关的说明文字，其格式为：

`Lexical error at Line [行号]: [说明文字].`

说明文字的内容没有具体要求（例如：非法字符），但是错误类型和出错的行号一定要正确，因为这是判断输出错误提示信息是否正确的唯一标准。

（3）系统的输入形式：要求能够通过文件导入测试用例。测试用例要涵盖“实验内容”中列出的各类单词。

（4）系统的输出形式：打印输出测试用例对应的 token 序列。

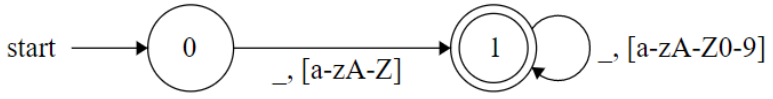
二、文法设计

得分

1. **标识符**。正则表达式为

`(_[a-zA-Z])(_[a-zA-Z][0-9])*`

DFA 状态转移图为



2. **关键字**。关键字有 `if`，`else`，`while`，`do`，`struct`，`return`，`true`，`false`；数据类

型有 int，float，bool 和 string。

3. **运算符**。支持算术运算（加法+，减法-，乘法*，除法/，模%）、关系运算（等于==，大于>，小于<，大于等于>=，小于等于<=，不等于!=）、逻辑运算（与&&，或||，非!）和位运算（与&，或|，非~，异或^）。
4. **界符**。支持赋值号=，分号; ，左方括号[，右方括号] ，左花括号{ ，右花括号} 。
5. **数字常数**。支持整数、浮点数、科学计数法、八进制数和十六进制数。
   十进制整数正则表达式为

   ```
   0|([1-9][0-9]*)
   ```

   八进制整数正则表达式为

   ```
   0[0-9]*
   ```

   十六进制整数正则表达式为

   ```
   0(x|X)[0-9|a-f|A-F]+
   ```

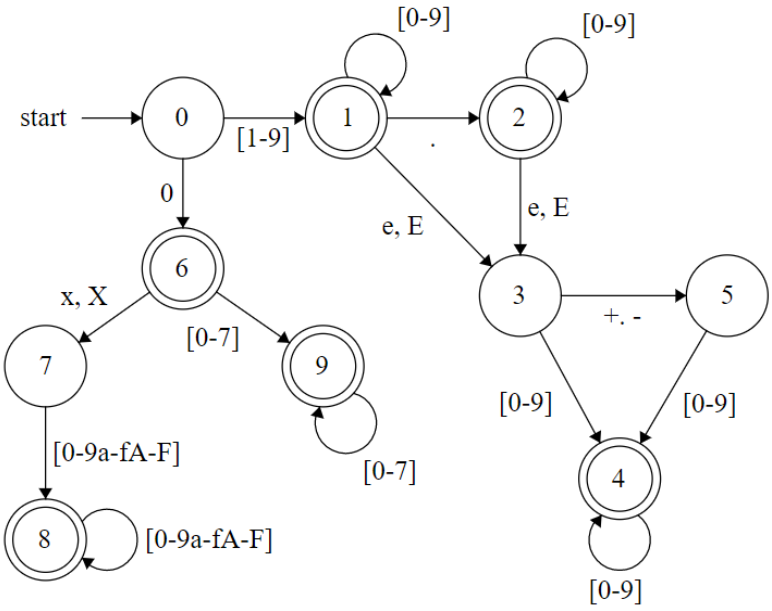
   一般浮点数正则表达式为

   ```
   [0-9]*.[0-9]+
   ```

   科学计数法正则表达式为

   ```
   [0-9]*.[0-9]+(e|E)[0-9]+
   ```

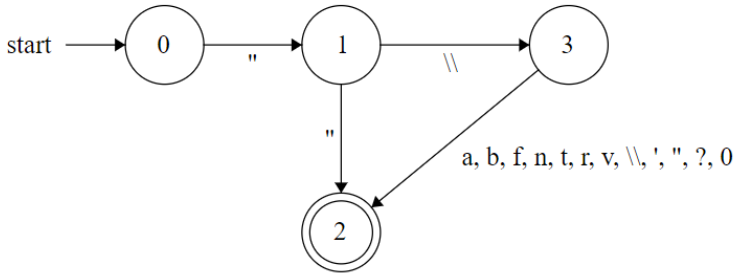
   将以上情况合成为一个 DFA，即得



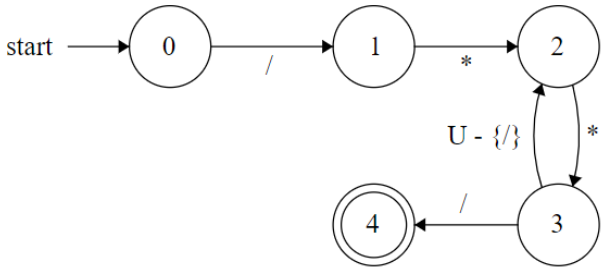
6. **字符串常量**。支持 ASCII 字符集中的可打印字符组成的字符串，支持反斜杠\开头的转义字符引用。正则表达式为

   ```
   "[(\\{转义字符}){非转义字符}]*"
   ```

   DFA 状态转移图为



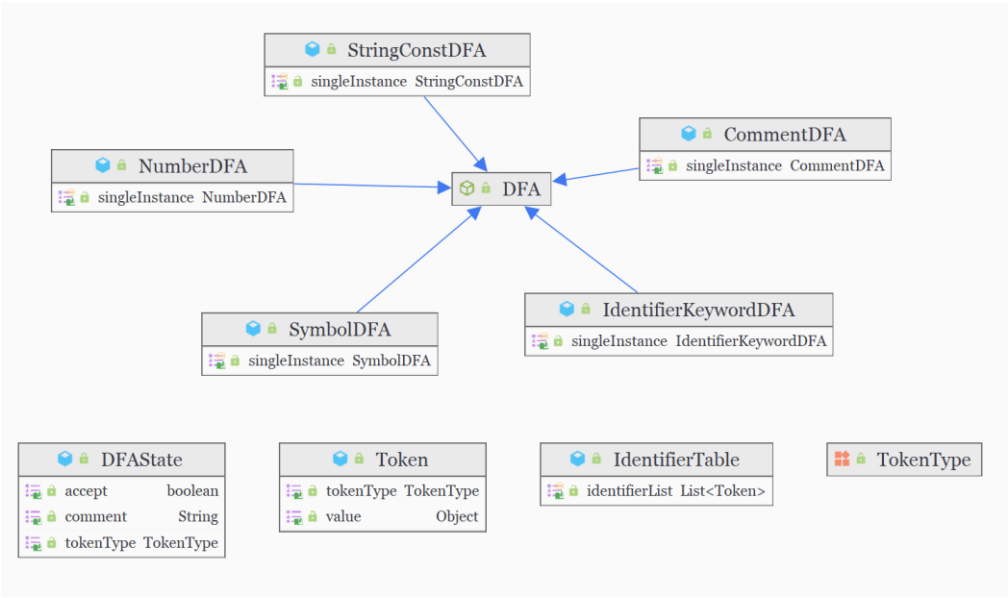
7. **注释**。支持 /*……*/ 形式的注释。DFA 状态转移图为



三、系统设计 | 得分

项目的 JavaDoc 文档在 `javadoc` 目录下（用浏览器打开 `javadoc/index.html` 即可）。

（1）**系统概要设计。**项目类图如下图所示



a) `package dfa` 中包含了 DFA 类、DFA 状态类和各个具体 DFA 类。
`DFA` 是一个抽象类，被 `NumberDFA`、`CommentDFA` 和 `SymbolDFA` 等具体 DFA 类继承。每个具体 DFA 类都包括若干个状态及状态之间的转移。这个包中各个类的说明如下图所示。

程序包 dfa

类概要

类	说明
CommentDFA	
DFA	有限状态自动机DFA类
DFAState	DFA状态
IdentifierKeywordDFA	标识符和关键字DFA。
NumberDFA	数字DFA（可处理十进制数、浮点数、科学计数法、八进制和十六进制数）。
StringConstDFA	字符串常量DFA。
SymbolDFA	各类符号DFA。

b) `package token` 中包含了 `Token` 类、Token 类别枚举类、标识符表类和一个辅助方法类。

`Token` 类实现了词法分析的最小单元 token，包括两个属性：token 类别和 token 值。标识符表类 `IdentifierTable` 用于储存和标记词法分析时所识别出的标识符。这个包中各个类的说明如下图所示。

程序包 token

类概要

类	说明
IdentifierTable	标识符表
Token	词法分析单元token
Utils	辅助方法和参数

枚举概要

枚举	说明
TokenType	token类型

（2）**系统详细设计。**

a) **核心数据结构设计。**

i. **`DFAState` 类。**一个 DFA 状态有如下属性：

- `walkMap`：存储转移边和后继状态
- `isAccept`：本状态是否是接受状态
- `tokenType`：以本状态结束的 token 的类别
- `legalEndChars`：合法终结字符集

这里的关键设计是 legalEndChars，它仅在接受状态中有效。它的含义是，“当且仅当下一个输入字符是该集合中的字符时，当前 token 能被合法地分割”。之所以要加入这个设计，是为了增加词法分析器的鲁棒性，并实现功能模块化

前提下的“尽早报错”。

例如，对于 `123.123.123` 这样一个非法输入，读入到第二个界符`.`时，如果不用合法终结字符的方法进行判断，词法分析器就会分割出浮点数 `123.123`，界符`.`和整数 `123` 共三个 token 并直接交付给语法分析器，不报错。这个流程虽然不违反词法分析器的功能设定，但终究没有做到“物尽其用”，延迟了报错时间。而如果用合法终结字符集的方法，先设定好数字 DFA 中浮点数状态（即之前数字 DFA 图中的状态 2）的合法终结符集（集合中当然不包括界符.），等到读入到第二个界符`.`的时候，DFA 就能判断出当前无法分割，直接报错。

该类的文档页面如下图所示。

程序包 dfa

类 DFAState

java.lang.Object
  dfa.DFAState

```
public class DFAState
extends java.lang.Object
```

DFA状态

构造器概要

构造器

构造器	说明
`DFAState(java.lang.String comment)`	构造一个DFA非接受状态
`DFAState(java.lang.String comment, java.lang.String legalEndChars)`	构造一个DFA接受状态，token类型定义为 TokenType.UNDETERMINED
`DFAState(java.lang.String comment, TokenType tokenType, java.lang.String legalEndChars)`	构造一个DFA接受状态

方法概要

所有方法 | 实例方法 | 具体方法

修饰符和类型	方法	说明
void	`addTransition(char c, DFAState state)`	添加转移边
void	`addTransition(java.lang.String string, DFAState state)`	添加转移边。
java.lang.String	`getComment()`	
DFAState	`getNext(char c)`	获取下一个状态
TokenType	`getTokenType()`	
boolean	`isAccept()`	
boolean	`isLegalEndChar(char ch)`	
java.lang.String	`toString()`	

ii. **DFA 类**。一个有穷状态自动机 DFA 类有如下属性：

- `startState`：开始状态
- `stateNumber`：状态数
- `states`：状态数组

DFA 类中用数组存储状态。该类的核心方法是 `walk`，原型为

```
public Pair<Integer, Token> walk(String line, int startPos)
```

它的作用是输入字符串，模拟 DFA 转移，匹配策略是贪心地尽量往后拓展。具体来说就是，从字符串 `line` 的第 `startPos` 个字符开始输入，DFA 从初始状态开始转移，在所达到的最远位置处分割为一个合法 token。然后，程序会调用 `analyze` 方法（用于分析该 token 的类型和值）。最后，`walk` 方法返回的就是分割位置和 `Token` 的二元组；如果匹配结果为失败，返回的 `Token` 为空（`null`）。

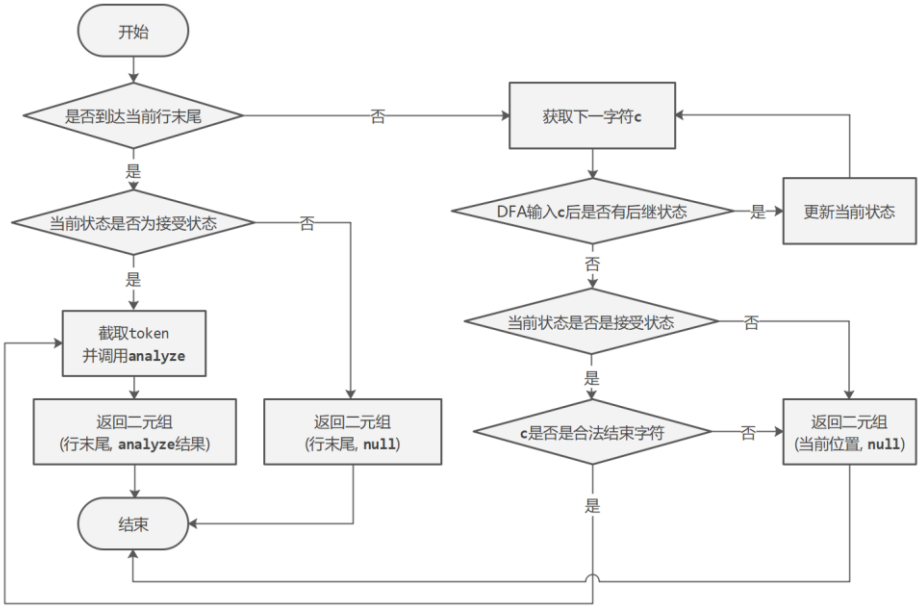
DFA 类的主要方法及说明如下表所示。

方法概要

所有方法 | 实例方法 | 抽象方法 | 具体方法

修饰符和类型	方法	说明
abstract `Token`	**`analyze`**(`DFAState` state, java.lang.String token)	根据自动机接受状态和token字符串分析token类型和值
abstract boolean	**`isLegalEndChar`**(char c, `DFAState` state)	判断字符是否是状态的合法结束输入
org.apache.commons.lang3.tuple.Pair<java.lang.Integer, `Token`>	**`walk`**(java.lang.String line, int startPos)	输入字符串，模拟DFA转移。

核心方法 `walk` 的流程图如下所示。

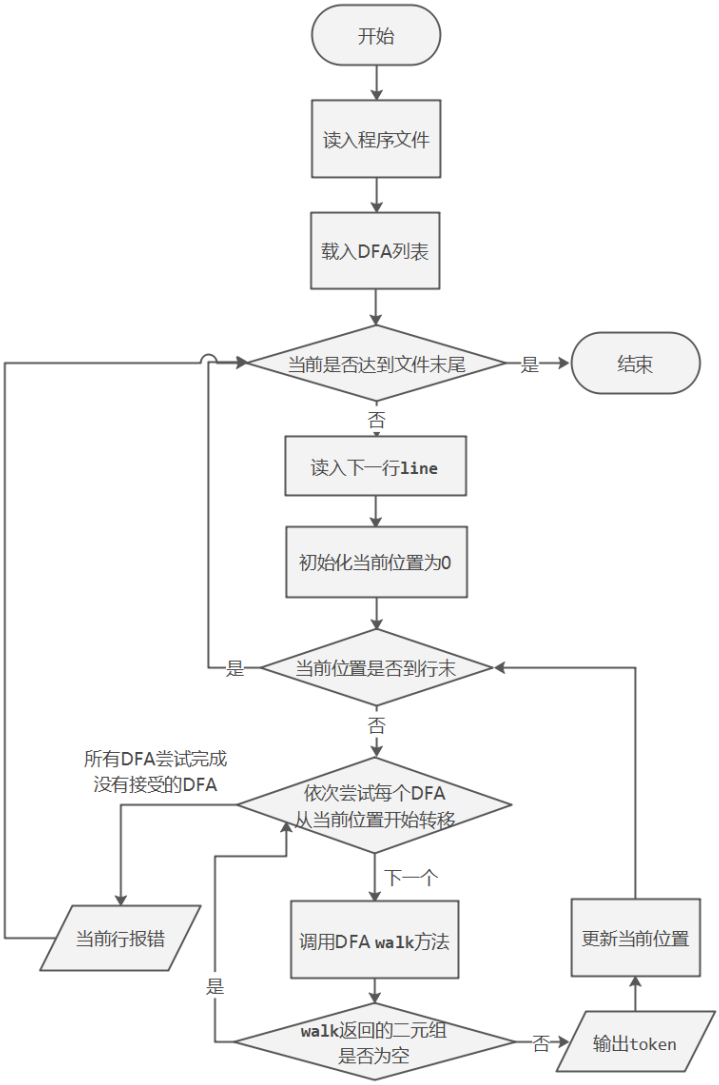


iii. `NumberDFA`、`SymbolDFA` 和 `CommentDFA` 等具体 DFA 类。它们继承 DFA 抽象类，构造自己的 DFA 状态和转移，设置初始状态，并重写基类的 `analyze` 等方法。

此外，为确保每个 DFA 都是唯一的，这些抽象类都采用了单例模式，即将构造方法的权限设置为 `private`，仅允许用静态方法 `getSingleInstance` 用于获取该类的唯一实例。

iv. `Main` 类中的 `main` 方法用于读入文件、调用 `package dfa` 中的接口进行词法分析并输出分析结果。

核心方法 `main` 的程序流程图如下所示。



四、系统实现及结果分析

得分

（1）遇到的最主要问题是**确定词法分析器的功能范围**。

例如，如果输入 123.123.123，词法分析器是否需要报错呢？我的考量是报错。因为这样完善了词法分析器的分词策略，而且提早了报错的时间。

但这样做也有缺点：一是增加了词法分析器和语法分析器的耦合性，二是增加了词法分析器的复杂度。权衡之后，我选择采用报错。如果在之后的实验中发现不如完全交给语法分析器，就删去这一功能。

（2）对于以下程序，

```
int main() {
    int x=sqrt(a[0x100]);
    float y = 123.123.123;
    float z = 1.23e+12;
    for (int i=0;i<=123;i=i+1){
        if(x==1)
            x=2;
        else
            while (true) x=3;
    }
    printf("Hello World!\n");
    return 0;
}
/* This is illegal comment *
/*This is legal comment.*/
```

词法分析器的输出结果如下表所示（顺序为第一列从上到下、第二列从上到下、第三列从上到下）。

(DT_INTEGER ,)	(KW_FOR ,)	(IDENTIFIER ,x)
(IDENTIFIER ,main)	(ROUND_LEFT ,)	(ASSIGNMENT ,)
(ROUND_LEFT ,)	(DT_INTEGER ,)	(CONST_INTEGER,2)
(ROUND_RIGHT ,)	(IDENTIFIER ,i)	(SEMICOLON ,)
(BEGIN ,)	(ASSIGNMENT ,)	(KW_ELSE ,)
(DT_INTEGER ,)	(CONST_INTEGER,0)	(KW_WHILE ,)
(IDENTIFIER ,x)	(SEMICOLON ,)	(ROUND_LEFT ,)
(ASSIGNMENT ,)	(IDENTIFIER ,i)	(CONST_BOOLEAN,)
(IDENTIFIER ,sqrt)	(LESS_EQUAL ,)	(ROUND_RIGHT ,)
(ROUND_LEFT ,)	(CONST_INTEGER,123)	(IDENTIFIER ,x)
(IDENTIFIER ,a)	(SEMICOLON ,)	(ASSIGNMENT ,)
(SQUARE_LEFT ,)	(IDENTIFIER ,i)	(CONST_INTEGER,3)
(CONST_INTEGER,256)	(ASSIGNMENT ,)	(SEMICOLON ,)
(SQUARE_RIGHT ,)	(IDENTIFIER ,i)	(END ,)
(ROUND_RIGHT ,)	(ADD ,)	(IDENTIFIER ,printf)
(SEMICOLON ,)	(CONST_INTEGER,1)	(ROUND_LEFT ,)
(DT_FLOAT ,)	(ROUND_RIGHT ,)	(CONST_STRING ,Hello World!\n)
(IDENTIFIER ,y)	(BEGIN ,)	(ROUND_RIGHT ,)
(ASSIGNMENT ,)	(KW_IF ,)	(SEMICOLON ,)
(DT_FLOAT ,)	(ROUND_LEFT ,)	(KW_RETURN ,)
(IDENTIFIER ,z)	(IDENTIFIER ,x)	(CONST_INTEGER,0)
(ASSIGNMENT ,)	(EQUAL ,)	(SEMICOLON ,)
(CONST_FLOAT ,1.22999996E12)	(CONST_INTEGER,1)	(END ,)
(SEMICOLON ,)	(ROUND_RIGHT ,)	(COMMENT ,This is legal comment.)

（3）对于同一程序，错误分析报告为

```
Lexical error at Line 3: Cannot analyze at position 11
Lexical error at Line 14: Cannot analyze at position 1
```

（4）程序输出和错误报告均符合预期。输出结果能够准确无误地实现分词并识别出 token 的类型和值；错误报告能够准确判断词法错误并给出错误位置。

指导教师评语：

日期：