## 第一阶段设计文档

第一阶段主要内容：GUI、词法分析

实现方案：

1. **工具**：Intellij IDEA
2. **环境**：JDK 1.8，操作系统Windows10和Mac OS
3. **技术**：编程语言采用Java，第一阶段主要完成GUI和词法分析的部分，使用面向对象的方式实现。
4. **词法分析**

首先给出CMM语言词法的**描述文法**如下：

G = {Vt,Vn,S,P}

Vt={ a,b,c…y,z,0,1,2…8,9,+,-,\*,/ ,>,=,<,;,(,),},{,[,] }

Vn={A, B…N}

P:

S—>(a|b|c…y|z)A**|**-C**|**(0|1|2…8|9)D**|**(+|>|(|)|{|}|[|]|;)G**|**=H**|**<J**|**/L**|**\*N A—> (a|b…y|z|0|1…8|9)A**|**\_B**|**nil

B--> (a|b…y|z|0|1…8|9|\_)B**|**(a|b…y|z|0|1…8|9)A

C—> (0|1…8|9)D**|**nil

D—> (0|1…8|9)D**|**\_E**|**nil

E--> (0|1…8|9)F

F--> (0|1…8|9)F**|**nil

G-->nil

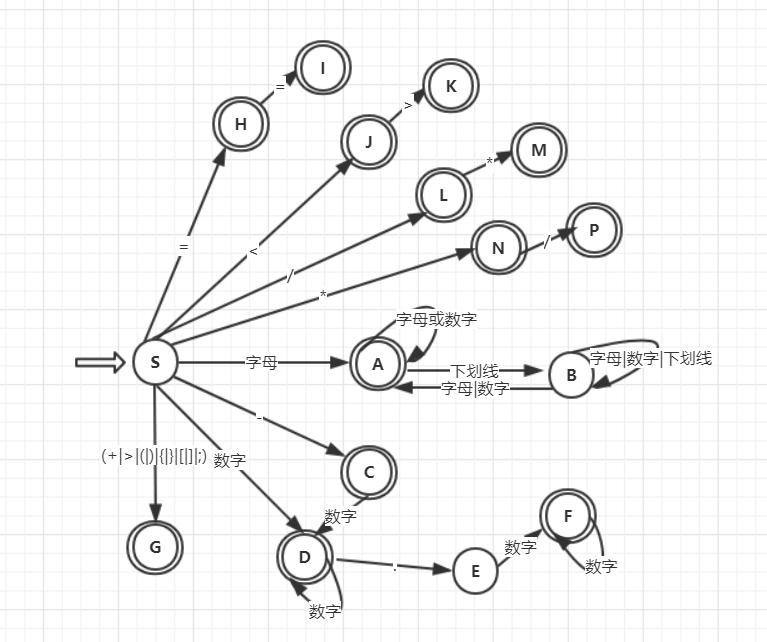
H-->=|nil

J-->>|nil

L-->\*|nil

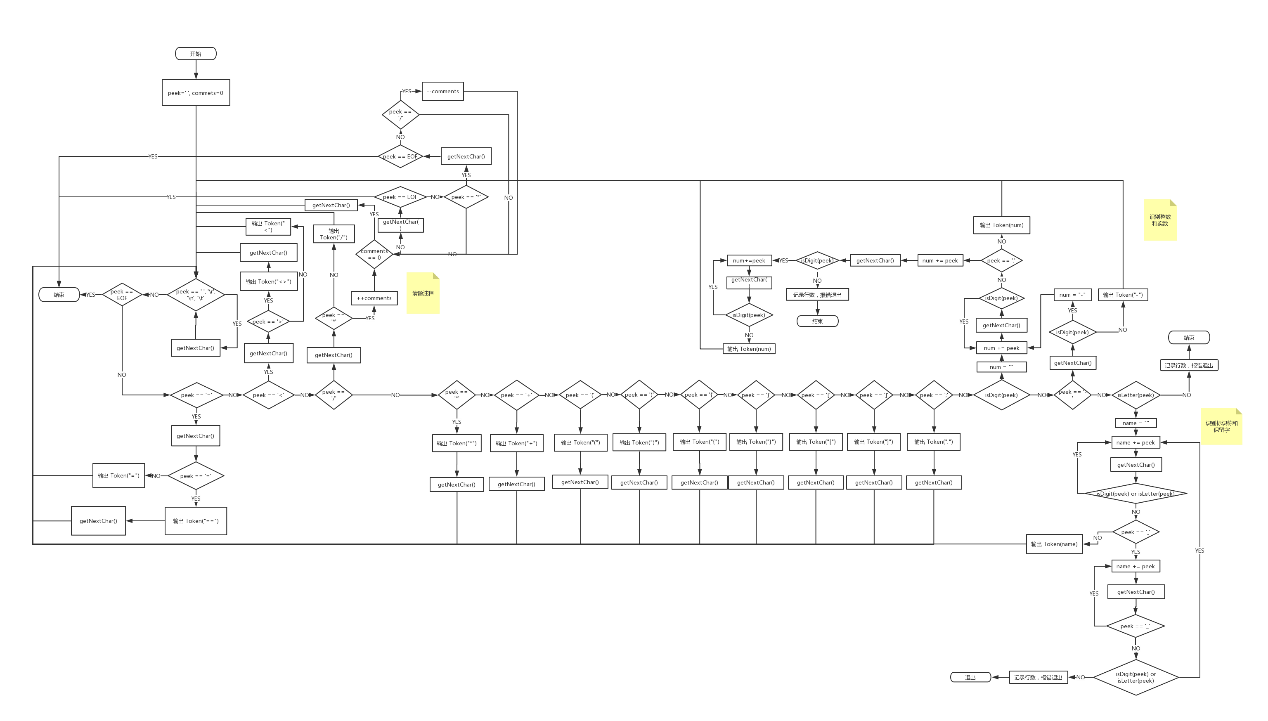
N-->/|nil

根据以上文法，画出其对应的有限状态机，以有限状态机为指导进行CMM语言的词法分析。**有限状态机的状态转移图**示如下：



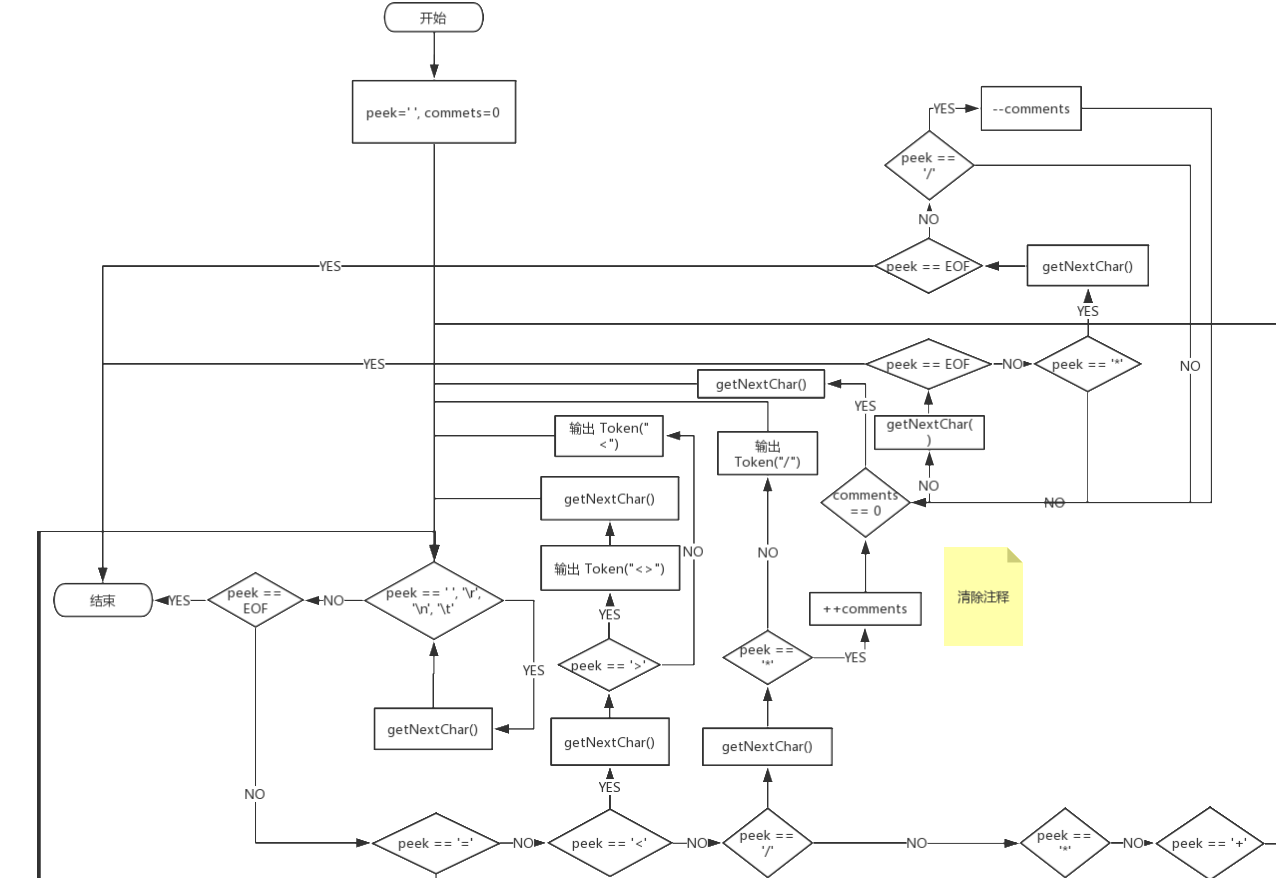
根据状态机，对核心的词法识别部分绘制如下的流程图，以此制导词法分析功能的实现。

**完整流程图**如下：

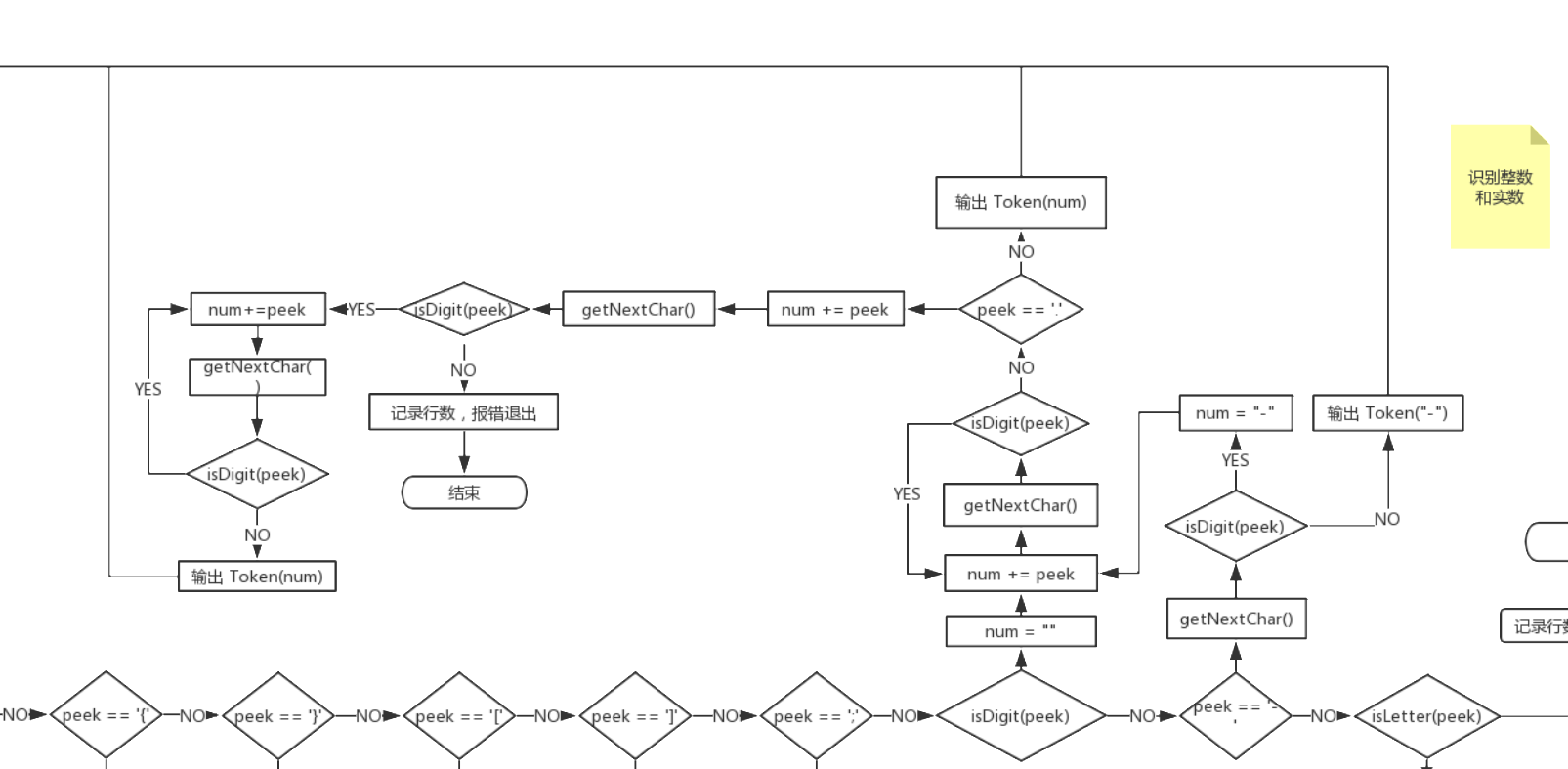


其中除单字符和双字符属性字的识别外，主要包含三个部分的流程，分别是注释的清除、整数及实数的识别、标识符的识别。

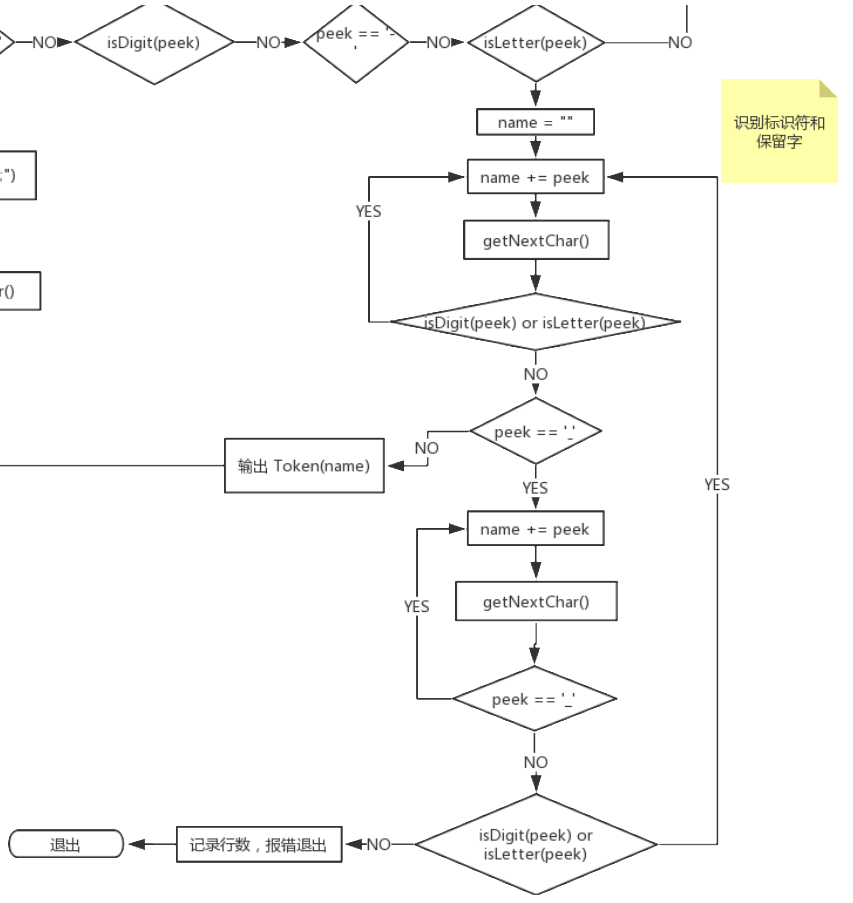
**清除注释部分的流程图**如下：



**整数和实数的识别流程**如下：



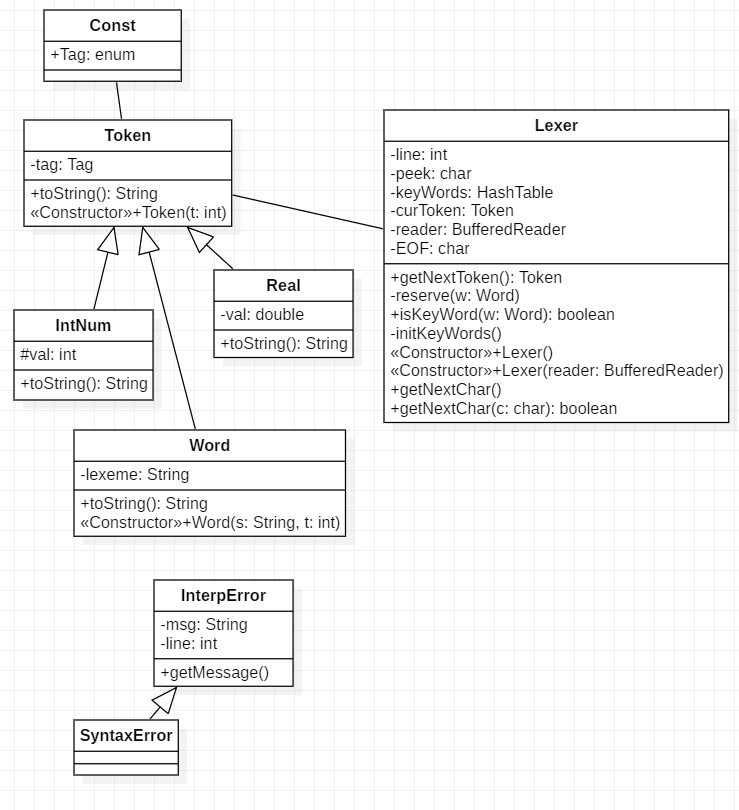
**标识符的识别流程**如下：



使用整数对属性字的类别进行编码，**编码表**如下：

|  |  |
| --- | --- |
| 单词符号 | 种别码 |
| if | 1 |
| else | 2 |
| int | 3 |
| while | 4 |
| read | 5 |
| write | 6 |
| real | 7 |
| + | 8 |
| - | 9 |
| \* | 10 |
| / | 11 |
| = | 12 |
| < | 13 |
| > | 14 |
| == | 15 |
| <> | 16 |
| ( | 17 |
| ) | 18 |
| ; | 19 |
| { | 20 |
| } | 21 |
| /\* | 22 |
| \*/ | 23 |
| [ | 24 |
| ] | 25 |
| 常数 | 49 |
| 标识符 | 50 |

基于面向对象的思想，为属性字建立一个基类，并对三种不同类别的属性字建立相应的子类，同时建立一个类专门存放单词的编码等常数。对于主要的词法分析部分，使用一个类表示词法分析器，专门管理词法分析的相关方法。最后，对于该过程可能产生的错误，建立相应的异常类进行表示。设计的类图如下：



相关说明：

Const：放置程序代码中使用的各个常数，包括token类别编码Tag

Token：属性字，包含属性tag表示其类别的编码

IntNum：整数的属性字，包含特殊属性val表示其具体值

Real：浮点数的属性字，包含特殊属性val表示其具体值

Word: 复合词法单元的属性字，包含特殊属性lexeme表示其词素的字符串

InterpError: 表示解释器执行过程，源码中发现的错误的异常类，是其他子异常的基类

SyntaxError：表示解释器识别到的语法错误（包含词法和语法）的异常

Lexer：词法分析器，用从文件中得到的BufferedReader输入流对象读取字符，并进行token的识别和生成。

* line: 当前行号
* peek: 当前读到的字符
* keyWords: 语言的保留字
* curToken：当前识别到的token
* reader：源文件的输入流对象
* EOF: 常数，文件尾字符的表示
* getNextToken：识别新的token，该过程会改变reader在文件中读到的位置，以及当前字符peek. 最后返回新的token，并为curToken赋值；

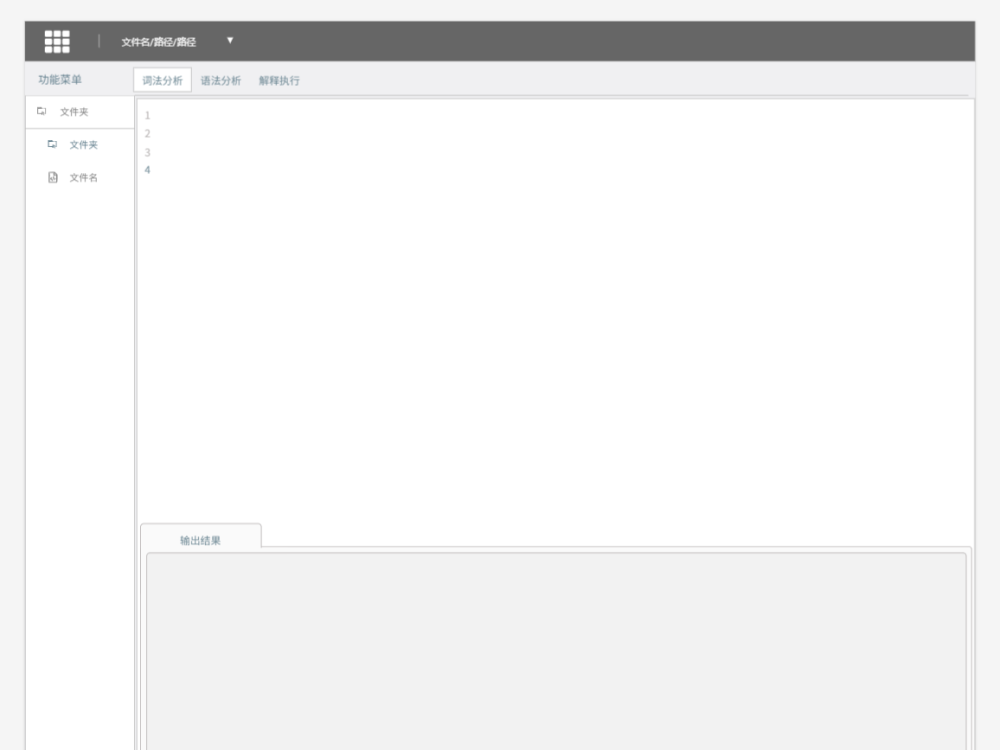
若返回的token为null, 则表明当前文件已到文件尾；

该过程可能抛出异常IOException和SyntaxError.

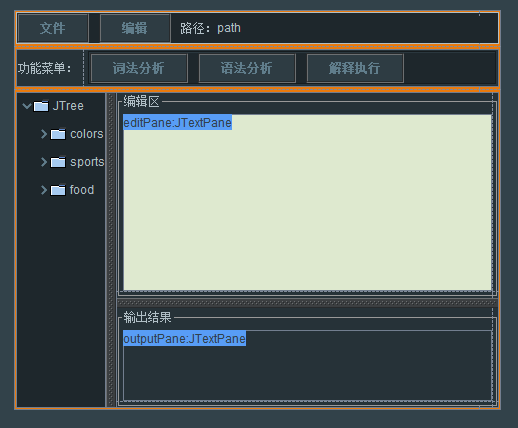
* reserve：将参数的属性字作为保留字记录在keyWords中
* isKeyWord：识别参数是否为语言的保留字
* initKeyWords: 初始化关键字
* getNextChar：从源文件中读取下一个字符，改变peek，若达文件尾，置peek为EOF
* getNextChar(c)：从源文件中读取下一个字符，并比较是否等于 c ，此过程改变peek，返回比较结果.

1. **GUI**

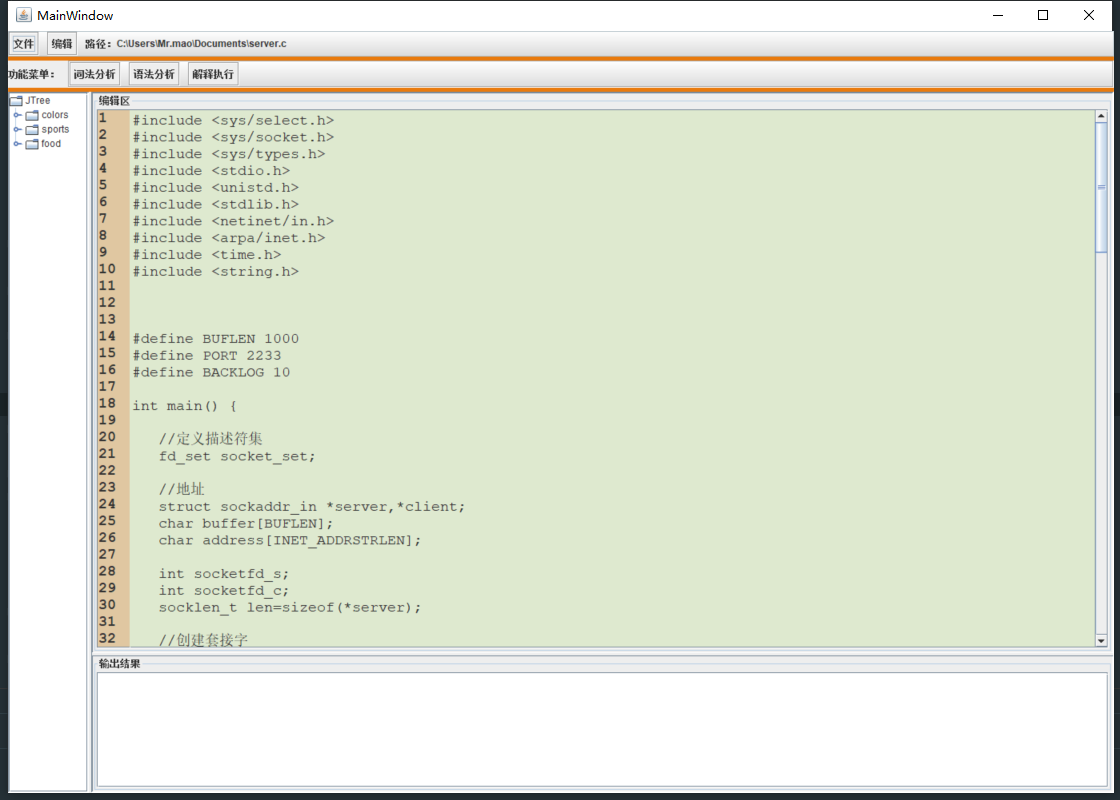
参考市面上常见的IDE，如Visual Studio，IDEA，Pycharm，使用墨刀设计出界面原型如下：



使用IDEA的界面设计器设计出蓝图：



具体展示效果如下图所示：

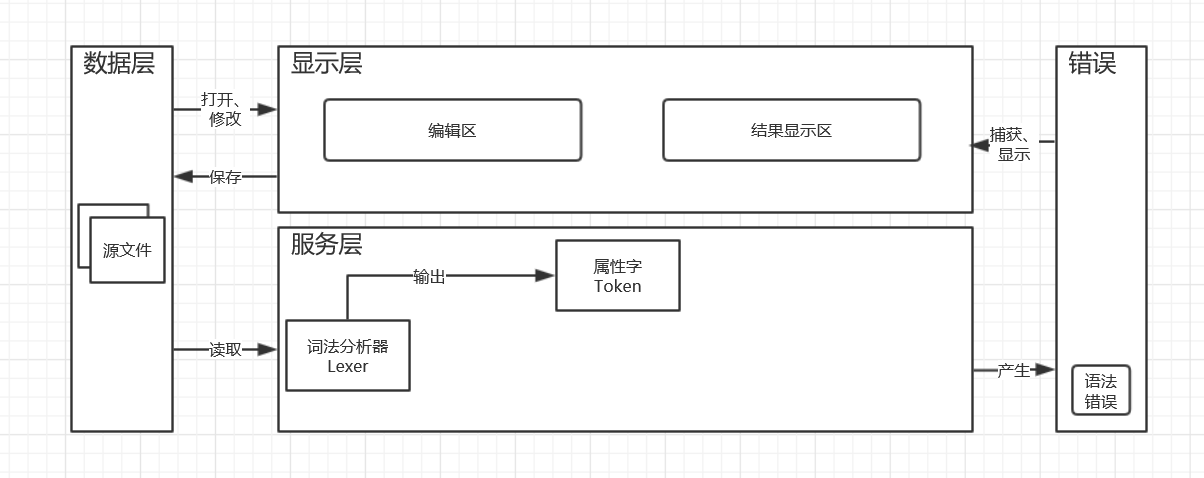


功能实现难点：编辑区行号的设置

实现方案1：设置两个个JTextPane，一个用来显示行号，一个作为编辑区。监听编辑区的键盘事件，每次在编辑区按键时都会刷新行号，同时要设置行号区域的滚动条隐藏且与编辑区的保持一致。问题是，这样导致在编辑区输入过快时，行号刷新不及时，有重影。

实现方案2：只使用一个JTextPane，作为编辑区，使用JSrcollPane的setRowHeaderView方法，用一个list来记录行号，每次键入时比较编辑区行大小与记录的，依此从list中添加或删除元素，不用每次都刷新整个行号，有更好的显示效果。

1. **架构**



在第一阶段的设计中，该系统共分为三个层次：数据层，显示层和服务层。

**数据层**为用户编写的源文件。

**显示层**通过GUI的操作对源文件进行打开和修改，并将修改保存回文件。

**服务层**主要是核心的解释器功能，第一阶段中暂时只包含词法分析器部分。服务层也是从数据层中读取文件，并分析其中的CMM代码进行词法分析，生成对应于各个单词的属性字Token。过程中可能产生相应的错误，此时停止源码分析，立即将错误抛出。

当显示层调用服务层的功能时，将当前文件修改保存回源文件，并将源文件交由服务层进行词法分析。服务层将生成的属性字返回给显示层，并在结果显示区中显示。当识别到某个错误时，显示层应捕获该错误并将错误信息输出在结果显示区中。