程序设计实践实验报告

一种领域特定脚本语言的解释器的设计与实现

姓名：倪玮昊

学号：2020211346

班级：2020211304

日期：2022/11/19

1 概述

1.1实验内容

定义一个领域特定脚本语言，这个语言能够描述在线客服机器人（机器人客服是目前提升客服效率的重要技术，在银行、通信和商务等领域的复杂信息系统中有广泛的应用）的自动应答逻辑，并设计实现一个解释器解释执行这个脚本，可以根据用户的不同输入，根据脚本的逻辑设计给出相应的应答。

大作业

本作业考察学生规范编写代码、合理设计程序、解决工程问题等方面的综合能力。满分100分，具体评分标准如下：

风格：

满分15分，其中代码注释6分，命名6分，其它3分。

设计和实现：

满分30分，其中数据结构7分，模块划分7分，功能8分，文档8分。

接口：

满分15分，其中程序间接口8分，人机接口7分。

测试：

满分30分，测试桩15分，自动测试脚本15分

记法：

满分10分，文档中对此脚本语言的语法的准确描述。

注意：抄袭或有意被抄袭均为0分

学习时长 00:00:00

1.2 实验环境

IDEA 2022，Windows11 64 位

2 总体设计

主要设计

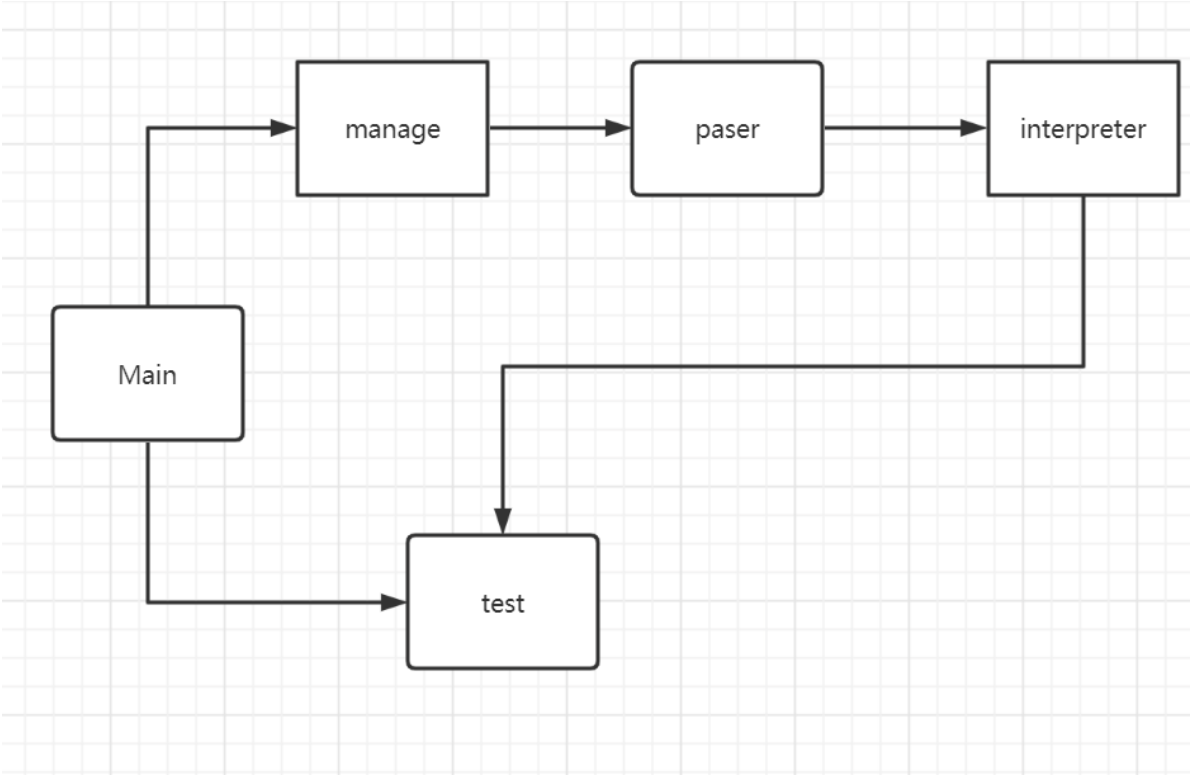
Main模块构造了主界,分别引向语法分析和测试界面

manage负责处理文法的输入删除和选择

paser 分析 test.txt 中自己设计的脚本语言。,根据脚本语言生成语法树,判断语法是否符合要求,如果错误则返回报错信息,如果正确则将词语储存到数据结构中。

interpreter 根据语法树翻译函数过程，将语言分析结果传给test模块。

test通过传输用户语句给interpreter来进行通话,同时构造聊天区的图形界面。



2.1 *Main* 模块的设计

part1:UI

Main模块构造了主界面,分别引向语法分析和测试界面

```
c = f.getContentPane();// 容器c嵌入窗体f中
c.setLayout(new FlowLayout());// 布局，流布局，flow（流动）
f.setTitle("自动客服机器人");
f.setBounds(600, 150, 400, 300);
f.setLocationRelativeTo(null);
JButton jb = new JButton("测试");
JButton jb3 = new JButton("管理文法");
ImageIcon imgicon = new ImageIcon("1.png");//图片插入，更改图片路径，需要注意后缀
JLabel imgjla = new JLabel(imgicon);
Dimension dim = new Dimension(360,203);
imgjla.setPreferredSize(dim);
c.add(imgjla);
c.add(jb);
c.add(jb3);
f.setResizable(false);
f.setVisible(true);
```

```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

2.2 manage模块的设计

part1:UI

```
c = f.getContentPane();// 容器c嵌入窗体f中
c.setLayout(new FlowLayout());// 布局，流布局，flow（流动）
f.setTitle("修改文法导入");
f.setBounds(600, 150, 400, 700);
f.setLocationRelativeTo(null);
//聊天记录显示区
ta = new JTextArea(10, 40);
tb = new JTextArea(25, 40);
//聊天记录输入区
ta.setEditable(false);
JTextField tf = new JTextField(40);
tf.requestFocus();//guang biao
JButton jb = new JButton("导入文法");
JButton jb1 = new JButton("删除文法");
JButton jb2 = new JButton("查看+选择文法");
JScrollPane p=new JScrollPane(ta);
c.add(p, BorderLayout.CENTER);
p=new JScrollPane(tb);
c.add(p, BorderLayout.CENTER);
c.add(tf, BorderLayout.SOUTH);
c.add(jb);
c.add(jb1);
c.add(jb2);
ta.setText("聊天内容"+"\n");
f.setResizable(false);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
```

part2:导入文法

```
chooser=new JFileChooser(".");
filter=new ExtensionFileFilter();
filter.addExtension("txt");//设置文件过滤器
chooser.addChoosableFileFilter(filter);
chooser.setAcceptAllFileFilterUsed(false);
int result=chooser.showOpenDialog(f);
if(result == JFileChooser.APPROVE_OPTION)
{
    String path = chooser.getSelectedFile().getAbsolutePath();//绝对路径
    File tempFile =new File( path.trim());//求文件名
    String fileName = tempFile.getName();
    if(isExist(fileName)){//检查文法是否已经存在
        tb.append("该文法已存在,请修改文件名"+'\n');
    }
    else{//若不存在则开始进行导入
        File source = new File(path);
        File dest = new File("grammer/"+fileName);
        try {
```

```
            copyFileUsingFileChannels(source, dest);
        } catch (IOException ex) {
            throw new RuntimeException(ex);
        }
        tb.append(fileName+"上传成功"+'\n');
        printallfile();
    }
```

part3:删除文法

```
String fileName = tf.getText();
if(isExist(fileName) && !tf.getText ().trim ().equals ("")){
    deleteFile(fileName);
    tb.append("该文法删除完成"+'\n');
    printallfile();

}
else{
    tb.append("该文法不存在"+'\n');
}
```

part4:选择文法

```
String fileName = tf.getText();
if(isExist(fileName) && !tf.getText ().trim ().equals ("")){
    tb.append(fileName+"内容为:"+'\n');
    try {
        readFile("grammer/"+fileName);
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
    File source = new File("grammer/"+fileName);
    File dest = new File("current.txt");

    int jud= 0;
    try {
        jud = p1.ParseFile("grammer/"+fileName);
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
    if(jud==0) {
        tb.append("选择该文法成功!" + '\n');
        try {
            copyFileUsingFileChannels(source, dest);//把文件拷贝到current.txt
        } catch (IOException ex) {
            throw new RuntimeException(ex);
        }
    }
    else{
        tb.append("选择该文法失败!" + '\n');
        judge(jud);//判断失败原因
    }
}
else{
```

```
    tb.append("文件"+fileName+" 不存在"+'\n');
}
```

part5:文件选择器

```
static class ExtensionFileFilter extends FileFilter {

    private ArrayList<String> extensions=new ArrayList<>();
    //自定义方法，用于添加文件后缀名
    public void addExtension(String extension) {
        if(!extension.startsWith("."))
            extension="."+extension;
        extensions.add(extension.toLowerCase());
    }
    public boolean accept(File file) {
        if(file.isDirectory()) return true;
        String name=file.getName().toLowerCase();
        for(String extension:extensions) {
            if(name.endsWith(extension)) return true;
        }
        return false;
    }
    @Override
    public String getDescription() {
        return null;

    }
}
```

part6:辅助函数

```
void judge(int i)//根据parser传回来的数据来输出错误原因,后两位为错误类型,前面为行数
boolean isExist(String name)//判断文法是否在电脑里已经存在
void deleteFile(String path)//删除文法
void readFile(String fileName) throws IOException //打印出选择的文法
void printallfile()//打印出已经拥有的文法
private static void copyFileUsingFileChannels(File source, File dest) throws
IOException //拷贝选择成功的代码到程序内
```

2.3 parser模块的设计

part1 整体阅读程序代码

```
clear();
if(getFileLineNum(fileName)>1000) return 10;//行数过多
File file = new File(fileName);
FileReader fr = new FileReader(file);
BufferedReader br = new BufferedReader(fr);
String line;
int count=0;
while((line = br.readLine()) != null){
    count++;
    String str=line.replace(" ","");
```

```
    // 一行一行地处理...
    String[] cut  = str.split(";");//用逗号分开每一个
    int error=ParseLine(cut);
    if(error!=0){
        clear();
        return count*100+error;//数字代表报错类型
    };
}
flag=true;
return 0;
```

part2 按每一行都关键词对代码进行分析

```
String[] cutdeeper  = cut[0].split("/");//读取第一行的关键字决定分析类型
if(cutdeeper[0].equals("Welcome")){
    return ParseWelcome(cut);
}
else if(cutdeeper[0].equals("Varchar")){
    return ParseVarchar(cut);
}
else if(cutdeeper[0].equals("Exit")){
    return ParseExit(cut);
}
else if(cutdeeper[0].equals("Unknown")){
    return ParseUnknown(cut);
}
else if(cutdeeper[0].equals("Branch")){
    return ParseBranch(cut);
}
else if(cutdeeper[0].equals("Remark") || cutdeeper[0].equals("")){
    return 0;
}
else if(cutdeeper[0].equals("Run") ){
    return ParseRun(cut);
}

else {
    return 1;
}
```

part3 关键词的处理

```
Boolean flag=false;//文法导入是否成功
ArrayList<String> branch =new ArrayList<String>();//记录每一个分支的处理手段
int numBranch=0;//分支数
ArrayList<String> Exit =new ArrayList<String>();//记录离开时的处理方式
int numExit=0;//结束数
ArrayList<String> Unknown =new ArrayList<String>();//记录未知字符的处理方式
int numUnknown=0;//未知情况处理方式数
String[][]Varchar=new String[1000][1000];//记录变量名和其实际意义
String []Process=new String[1000];//记录函数名
String []Branch=new String[1000];//记录分支的关键字
ArrayList<String> Run =new ArrayList<String>();//装入函数操作
int numVarchar=0;//记录变量数
```

```
ArrayList<String> Welcome =new ArrayList<String>();//转入欢迎操作
int numWelcome=0;//记录欢迎数
int numProcess=0;//记录函数数量
ArrayList<String> Key = new ArrayList<String>(Arrays.asList("Welcome", "Branch",
"Exit","Remark","Speak","Unknown","Process","Run","Varchar","LM1OWCC"));
//关键字列表
```

辅助函数

```
public static int getFileLineNum(String filePath) //检查行数
void clear()//导入失败时清除数据
boolean isKey(String test)//检查是不是关键字
int analyzeVar(String test)//检查变量是否已经声明
Boolean judgeProcess(String test)//判断函数名是否已经被声明了
int ParseRun(String cut[])//对Run开头的行进行分析
boolean judgeBranch(String test)//判断分支关键词是否被重复使用
int ParseBranch(String cut[])//对Branch开头的行进行分析
int ParseWelcome(String cut[])//对welcome开头的行进行分析
int ParseVarchar(String cut[])//对Varchar开头的行进行分析
public static boolean isNumeric1(String str)//判断是不是数字
int ParseExit(String cut[])//对Exit开头的行进行分析
int ParseUnknown(String cut[])//对Unknown开头的行进行分析
```

2.4 test模块的设计

part1 UI

```
// Content(包含)，pane（窗格）
c = f.getContentPane();// 容器c嵌入窗体f中
c.setLayout(new FlowLayout());// 布局，流布局，flow（流动）
f.setTitle("与客服聊天");
f.setBounds(600, 150, 400, 700);
f.setLocationRelativeTo(null);
//聊天记录显示区
ta = new JTextArea(35, 40);
//聊天记录输入区
ta.setEditable(false);
JTextField tf = new JTextField(40);
tf.requestFocus();//guang biao
JButton jb = new JButton("发送");
JButton jb1 = new JButton("导入测试桩");
JButton jb3 = new JButton("重置");
JScrollPane p=new JScrollPane(ta);
c.add(p, BorderLayout.CENTER);
c.add(tf, BorderLayout.SOUTH);
c.add(jb);
c.add(jb1);
c.add(jb3);
ta.setText("聊天内容"+"\n");
f.setResizable(false);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
```

part2发送消息

```
String str = tf.getText();
ta.append("我:"+str+"\n");
tf.setText("");
analyseWord(str);
```

part3 导入测试桩

```
chooser=new JFileChooser(".");
filter=new ExtensionFileFilter();
filter.addExtension("txt");
chooser.addChoosableFileFilter(filter);
chooser.setAcceptAllFileFilterUsed(false);
int result=chooser.showOpenDialog(f);
if(result == JFileChooser.APPROVE_OPTION)
{
    String path = chooser.getSelectedFile().getAbsolutePath();//绝对路径
    File source = new File(path);
    File dest = new File("test.txt");
    try {
        copyFileUsingFileChannels(source, dest);
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
    ta.append("测试桩导入成功"+'\n');
    analyseTestdoc();
}
```

part4 重置语法时刷新

```
ta.setText("");
tf.setText("");
ta.setText("聊天内容"+"\n");
interp=new interpreter(p1)
```

part5 时间线程,可以记录多久以后结束通话

```
static class Count extends Thread {
    int time=0;
    int limit;

    Count(int lim){
        time=0;
        limit=lim;
    }
    public void run() {
        while (true) {
            try {
                this.sleep(1000);
                time++;
                if(time>limit && Flag==1){
                    printTime();
```

```
                    ta.append("机器人:");
                    interp.interpreterExit();
                    Flag=0;
                    ta.append("对话结束.");
                    break;
                }
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }

        }
    }
```

part6 文件选择器

```
private ArrayList<String> extensions=new ArrayList<>();
//自定义方法，用于添加文件后缀名
public void addExtension(String extension) {
    if(!extension.startsWith("."))
        extension="."+extension;
    extensions.add(extension.toLowerCase());
}
public boolean accept(File file) {
    if(file.isDirectory()) return true;
    String name=file.getName().toLowerCase();
    for(String extension:extensions) {
        if(name.endsWith(extension)) return true;
    }
    return false;
}
@Override
public String getDescription() {
    return null;
}
```

part7 删除测试桩

```
String fileName = tf.getText();
if(isExist(fileName) && !tf.getText ().trim ().equals ("")){
    deleteFile(fileName);
    ta.append("该测试桩删除完成"+'\n');
    printallfile();

}
else{
    ta.append("该测试桩不存在"+'\n');
}
```

part8 自动测试

```
File folder = new File("Test");
File[] files = folder.listFiles();
for(File file:files) {
    try {
        fileTest("Test/"+file.getName());//将文件名导入该函数分析
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
}
```

part9 辅助函数

```
void analyseWord(String word)//分析每一句话,传给interpreter
static void printTime()//打印时间
void analyseTestdoc()//分析测试桩
private static void copyFileUsingFileChannels(File source, File dest)//拷贝测试桩进程序内部
void fileTest(String path) throws IOException //测试文档,按行读取
void deleteFile(String path)//删除测试桩
void printallfile()//打印出已经拥有的测试桩
boolean isExist(String name){//判断测试桩是否在电脑里已经存在
```

2.5 interpreter的设计

part1 复制parser中的数据

```
branch = (ArrayList<String>)p1.branch.clone();
Exit = (ArrayList<String>)p1.Exit.clone();
Unknown = (ArrayList<String>)p1.Unknown.clone();
Run = (ArrayList<String>)p1.Run.clone();
Welcome = (ArrayList<String>)p1.Welcome.clone();
Varchar=new String[1000][1000];//拷贝变量
for(int i = 0;i < p1.Varchar.length;i++){
    for(int j = 0;j < p1.Varchar[i].length;j++){
        Varchar[i][j]=p1.Varchar[i][j];
    }
}
```

part2 辅助函数

```
void interpreterWelcome()//分析Welcome开头语句
void interpreterUnknown()//分析Unknown开头语句
void interpreterWord(String word)//分析Branch开头语句
int getExittime()//返回设定的最迟回复时间
void interpreterExit()//分析Exit开头语句
void interpreterProcess(String name)//分析Run开头的语句
int interpreterSpeak(ArrayList<String> sentence,int i)//分析Speak开头语句
void interpreterVar(String var)//分析Varchar开头语句
```

3 语法设计说明

1、首先，我们设计出了一个自己的脚本，我们需要通过 parse 解析脚本，构建脚本语法树。

接下来我们需要设计如何实现构建语法分析树：通过函读取文件，在文件中一行一行读取，并忽略行尾行首的空白，忽略空行.Remark作为注释符,遇到Remark直接跳过分析

在我设计的语法中

```
 Branch / "投诉" ;Process/ complainProc ;Speak/"您的意见是我们改进工作的动力"/"请问您还有什么补充?";
```

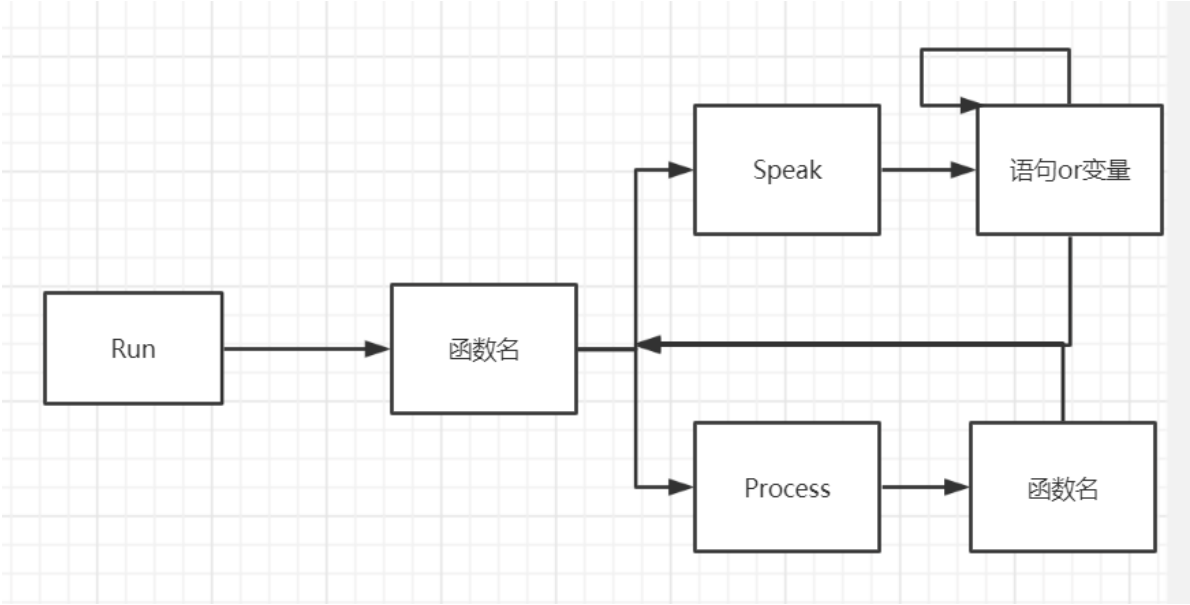
用/来分割同一句话中的不同词语,用;来分割不同的句子,所以通过对;和/的切割可以得到一个一个字符,进行录入和正确性的判断

同一行的句子的性质由开头的关键词决定,所以整个句子(在前继过程中已经被;切割过)的分析过程被分为:

1.Run

```
int len=cut.length;//计算被;切割成多少块
        for(int i=0;i<len;i++){
            String[] cutdeeper  = cut[i].split("/");//再次切割符号+.进行遍历
            for(int m=0;m<cutdeeper.length;m++){
                if(cutdeeper[m].equals("Run")) {//若为Run,长度必为2,Run+函数名
                    if (cutdeeper.length != 2 ) return 11;//返回错误
                    else {
                        Run.add(cutdeeper[m]);
                        m++;
                        if(!judgeProcess(cutdeeper[m])){//判断函数名有没有重复定义或为关键词
                            Run.add(cutdeeper[m]);
                            Process[numProcess]=cutdeeper[m];
                            numProcess++;
                        }
                        else {
                            return 17;//返回错误
                        }
                    }
                }
                else if(cutdeeper[m].equals("Process")){//函数中嵌套函数
                    if(cutdeeper.length!=2) return 3;//若为Process,长度必为2,Process+函数名
                    else {
                        Run.add(cutdeeper[m]);
                        m++;
                        if(judgeProcess(cutdeeper[m])) {
                            Run.add(cutdeeper[m]);
                        }
                        else return 11;
                    }

                }
```

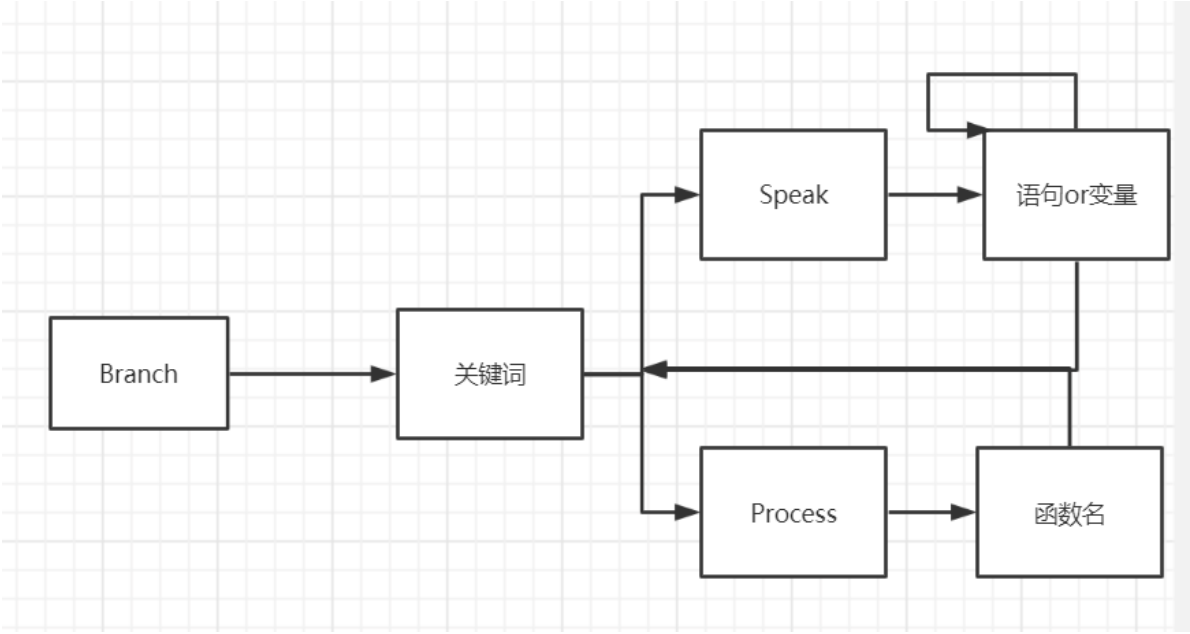
```
                else if(cutdeeper[m].equals("Speak")){//回答
                    if(cutdeeper.length<2) return 18;//回答后面必须有内容
                    Run.add(cutdeeper[m]);
                    m++;
                    if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                        if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                            Run.add(cutdeeper[m]);
                        }
                        else return 5;
                    }
                    else{
                        if(analyzeVar(cutdeeper[m])==-1) {//变量是否已经被定义
                            ;
                            return 4;
                        }
                        else{
                            Run.add(cutdeeper[m]);
                        }
                    }


                }
                else if(m!=0){//可能是第二个参数
                    if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                        if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                            Run.add(cutdeeper[m]);
                        }
                        else return 5;
                    }
                    else{
                        if(analyzeVar(cutdeeper[m])==-1) {
                            ;
                            return 4;
                        }
                        else{
                            Run.add(cutdeeper[m]);
                        }
                    }
                }
                else{
                    return 2;//非法字符
                }
            }

        }
        Run.add("LM1OWCC");//结束符
        return 0;
```



2.Branch

```
int len=cut.length;//看看被;分成多少段
        int Bflag=0;

        for(int i=0;i<len;i++){

            String[] cutdeeper  = cut[i].split("/");//再次切割符号/
            for(int m=0;m<cutdeeper.length;m++){
                if(i==0){
                    if(judgeBranch(cutdeeper[1])){
                        return 13;// branch中的关键词重复出现
                    }
                    else numBranch++;
                }
                if(cutdeeper[m].equals("Branch")){
                    if(Bflag==0) Bflag++;
                    else return 19;//同一句branch出现过多次
                    if(cutdeeper.length!=2) return 12;//branch格式错误
                    else {
                        branch.add(cutdeeper[m]);
                        m++;
                        branch.add(cutdeeper[m]);

                    }

                }
                else if(cutdeeper[m].equals("Process")){
                    if(cutdeeper.length!=2) return 3;//process后面只能跟着一个函数
                    else {
                        branch.add(cutdeeper[m]);
                        m++;
                        if(judgeProcess(cutdeeper[m])) {//判断process有没有被定义过
                            branch.add(cutdeeper[m]);
                        }
                        else return 11;
                    }
```

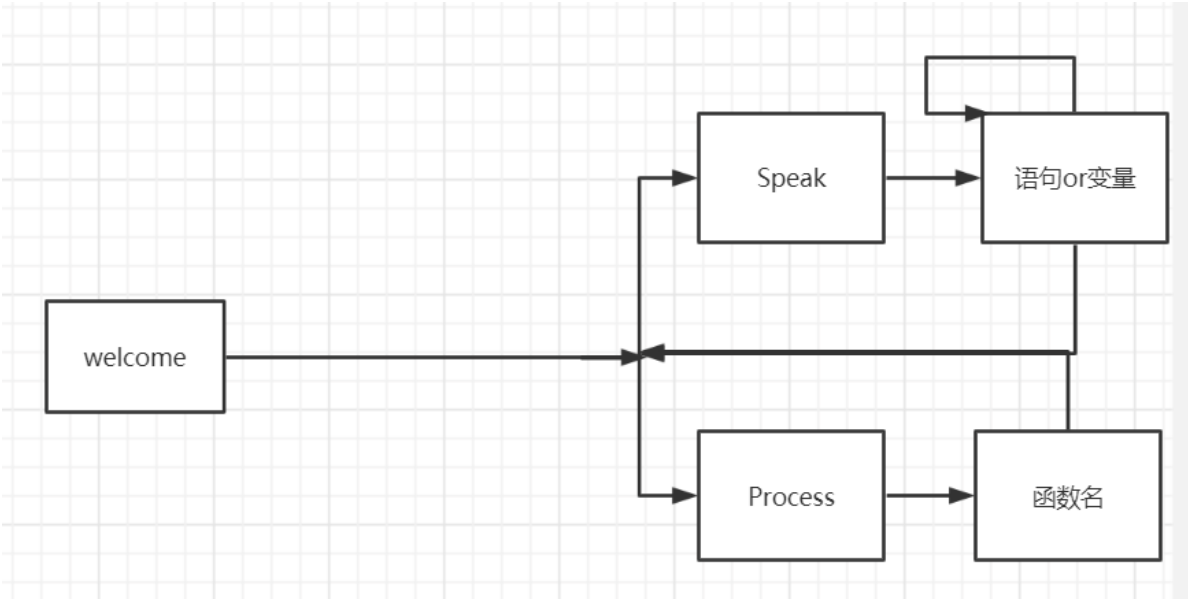
```
                }
                else if(cutdeeper[m].equals("Speak")){
                    if(cutdeeper.length<2) return 18;//speak 没有跟语句
                    branch.add(cutdeeper[m]);
                    m++;
                    if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                        if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                            branch.add(cutdeeper[m]);
                        }
                        else return 5;
                    }
                    else{
                        if(analyzeVar(cutdeeper[m])==-1) {

                            return 4;//变量未定义
                        }
                        else{
                            branch.add(cutdeeper[m]);
                        }
                    }


                }
                else if(m!=0){
                    if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                        if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                            branch.add(cutdeeper[m]);
                        }
                        else return 5;
                    }
                    else{
                        if(analyzeVar(cutdeeper[m])==-1) {//变量未定义

                            return 4;
                        }
                        else{
                            branch.add(cutdeeper[m]);
                        }
                    }
                }
                else{
                    return 2;
                }
            }

        }
        branch.add("LM10WCC");
        return 0;
```



3.Welcome

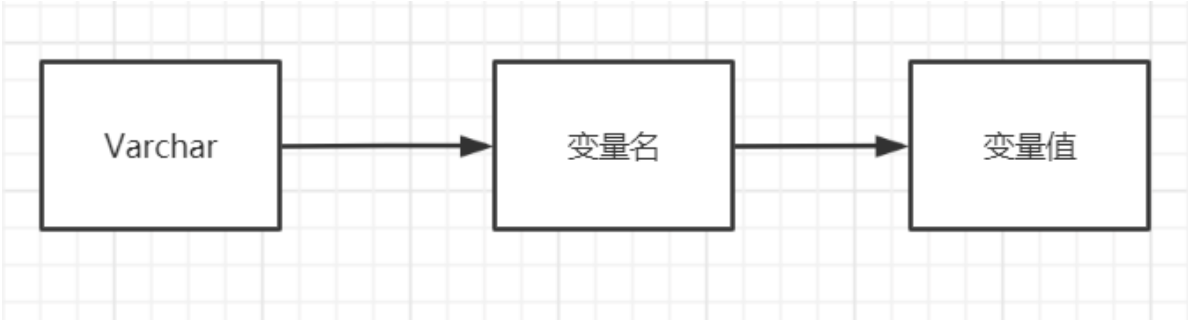
```
if(numWelcome==0) numWelcome++;//welcome只能出现一次
        else return 15;
        int len=cut.length;
        for(int i=0;i<len;i++){
            String[] cutdeeper  = cut[i].split("/");//再次切割符号+
            for(int m=0;m<cutdeeper.length;m++){
                if(cutdeeper[m].equals("Welcome")){
                    Welcome.add(cutdeeper[m]);

                }
                else if(cutdeeper[m].equals("Process")){
                    if(cutdeeper.length!=2) return 3;//Process后不只跟着一个函数
                    else {
                        Welcome.add(cutdeeper[m]);
                        m++;
                        if(judgeProcess(cutdeeper[m])) {
                            Welcome.add(cutdeeper[m]);
                        }
                        else return 11;
                    }

                }
                else if(cutdeeper[m].equals("Speak")){//
                    if(cutdeeper.length<2) return 18;//speak 没有跟语句
                    Welcome.add(cutdeeper[m]);
                    m++;
                    if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                        if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                            Welcome.add(cutdeeper[m]);
                        }
                        else return 5;
                    }
                    else{
                        if(analyzeVar(cutdeeper[m])==-1) {//变量未定义
                            return 4;
```

```
                    }
                    else{
                        Welcome.add(cutdeeper[m]);
                    }
                }


            }
            else if(m!=0){
                if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                    if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                        Welcome.add(cutdeeper[m]);
                    }
                    else return 5;
                }
                else{
                    if(analyzeVar(cutdeeper[m])==-1) {//变量未定义
                        return 4;
                    }
                    else{
                        Welcome.add(cutdeeper[m]);
                    }
                }
            }
            else{
                return 2;// 关键词错误
            }
        }

    }
    Welcome.add("LM1OWCC");
    return 0;
```



4.Varchar

```
int len=cut.length;
        for(int i=0;i<len;i++){
            String[] cutdeeper  = cut[i].split("/");//再次切割符号,
            for(int m=0;m<cutdeeper.length;m++){
                if(cutdeeper[m].equals("Varchar")){
                    if(cutdeeper.length!=3) return 6;//声明变量不规范
                    else if(analyzeVar(cutdeeper[1])==1) {
                        return 8;
                    }
                    else {
                        m++;
                        Varchar[numVarchar][0]=cutdeeper[m];
                        m++;
                        Varchar[numVarchar][1]=cutdeeper[m];
                        numVarchar++;
                    }

                }

            }

        }
        return 0;
```

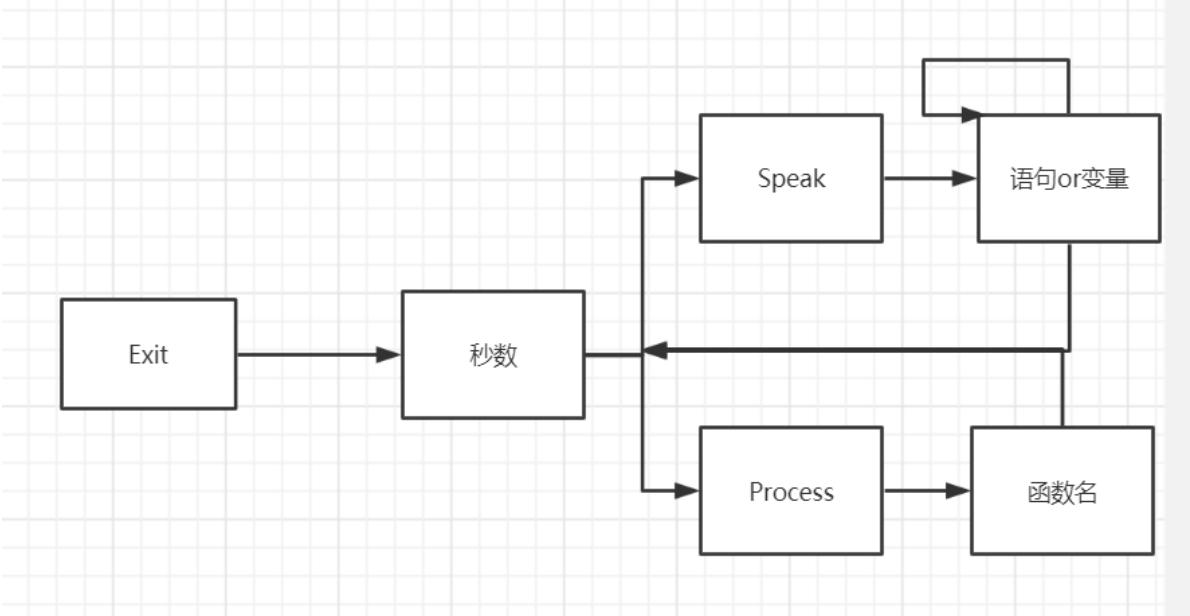


5.Exit

```
if(numExit==0) numExit++;//exit不能出现多次
        else return 16;
        int len=cut.length;
        for(int i=0;i<len;i++){
            String[] cutdeeper  = cut[i].split("/");//再次切割符号,
            for(int m=0;m<cutdeeper.length;m++){
                if(cutdeeper[m].equals("Exit")){
                    if(cutdeeper.length!=2) return 7;//exit格式错误
                    Exit.add(cutdeeper[m]);
                    m++;
                    if(!isNumeric1(cutdeeper[m])){//此处数字代表秒数,必须为整数
                        return 9;// exit跟着不是整数
                    }
                    else{
                        Exit.add(cutdeeper[m]);
                    }
                }
```

```
            else if(cutdeeper[m].equals("Process")){
                if(cutdeeper.length!=2) return 3;// process后不只跟着一个函数
                else {
                    Exit.add(cutdeeper[m]);
                    m++;
                    if(judgeProcess(cutdeeper[m])) {
                        Exit.add(cutdeeper[m]);
                    }
                    else return 11;// run后面的函数没有定义或格式错误
                }
            }
            else if(cutdeeper[m].equals("Speak")){
                if(cutdeeper.length<2) return 18;// speak 没有跟语句
                Exit.add(cutdeeper[m]);
                m++;
                if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                    if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                        Exit.add(cutdeeper[m]);
                    }
                    else return 5;//
                }
                else{
                    if(analyzeVar(cutdeeper[m])==-1) {//变量未定义
                        ;
                        return 4;
                    }
                    else{
                        Exit.add(cutdeeper[m]);
                    }
                }


            }
            else if(m!=0){
                if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                    if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                        Exit.add(cutdeeper[m]);
                    }
                    else return 5;//语句没用""包围
                }
                else{
                    if(analyzeVar(cutdeeper[m])==-1) {//变量未定义
                        ;
                        return 4;
                    }
                    else{
                        Exit.add(cutdeeper[m]);
                    }
                }
            }
            else{
                return 2;
            }
        }
```

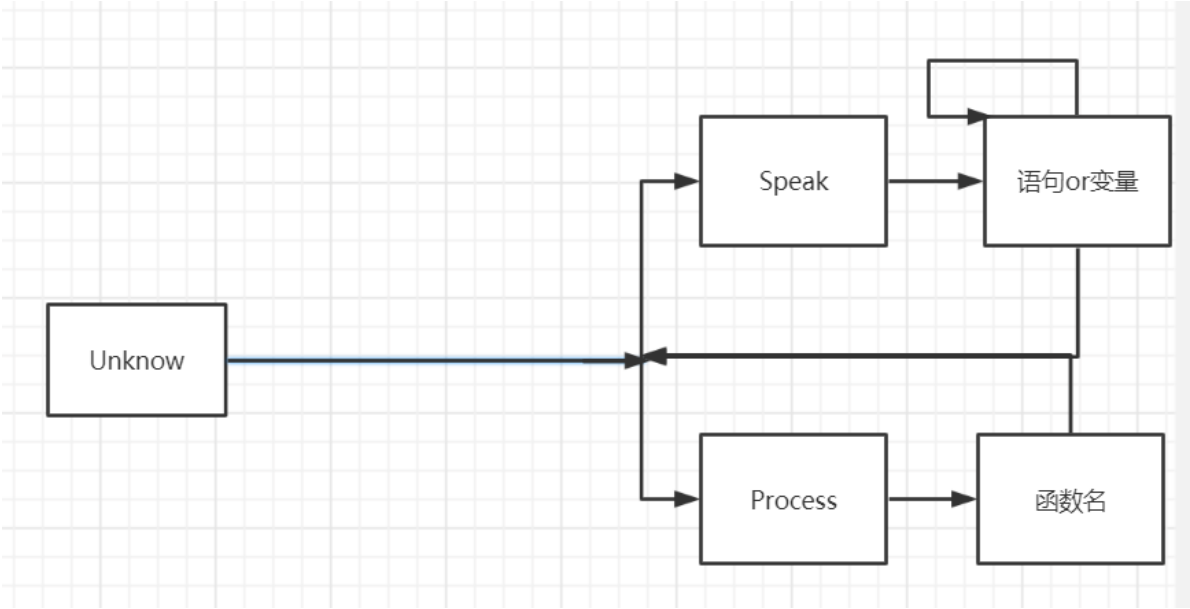
```
        }
        Exit.add("LM1OWCC");
        return 0;
```



6.Unknown

```
 //即不能匹配任何branch关键词的语句
        if(numUnknown==0) numUnknown++;//unknown不能出现多次
        else return 14;
        int len=cut.length;
        for(int i=0;i<len;i++){
            String[] cutdeeper  = cut[i].split("/");//再次切割符号,
            for(int m=0;m<cutdeeper.length;m++){
                if(cutdeeper[m].equals("Unknown")){
                    Unknown.add(cutdeeper[m]);

                }
                else if(cutdeeper[m].equals("Process")){//process后不只跟着一个函数
                    if(cutdeeper.length!=2) return 3;
                    else {
                        Unknown.add(cutdeeper[m]);
                        m++;
                        if(judgeProcess(cutdeeper[m])) {//run后面的函数没有定义或格式
错误
                            Unknown.add(cutdeeper[m]);
                        }
                        else return 11;
                    }

                }
                else if(cutdeeper[m].equals("Speak")){
                    if(cutdeeper.length<2) return 18;//speak 没有跟语句
                    Unknown.add(cutdeeper[m]);
                    m++;
                    if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                        if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                            Unknown.add(cutdeeper[m]);
                        }
```

```
                    else return 5;// 语句没用""包围
                }
                else{
                    if(analyzeVar(cutdeeper[m])==-1) {
                        ;
                        return 4;//变量未定义
                    }
                    else{
                        Unknown.add(cutdeeper[m]);
                    }
                }


            }
            else if(m!=0){
                if(cutdeeper[m].charAt(0)=='\"'){//判断语句是不是首尾都有"
                    if(cutdeeper[m].charAt(cutdeeper[m].length()-1)=='\"'){
                        Unknown.add(cutdeeper[m]);
                    }
                    else return 5;
                }
                else{
                    if(analyzeVar(cutdeeper[m])==-1) {//变量未定义

                        return 4;
                    }
                    else{
                        Unknown.add(cutdeeper[m]);
                    }
                }
            }
            else{
                return 2;
            }
        }

    }
    Unknown.add("LM1OWCC");
    return 0;
```



报错信息大全

错误类型 10 行数过多

错误类型 1 非法字符

错误类型 2 关键词错误

错误类型 3 process后不只跟着一个函数

错误类型 4 变量未定义

错误类型 5 语句没用""包围

错误类型 6 声明变量不规范

错误类型 8 变量已存在或和关键词冲突

错误类型 7 exit格式错误

错误类型 9 exit跟着不是整数

错误类型 11 run后面的函数没有定义或格式错误

错误类型 12 branch格式错误

错误类型 13 branch中的关键词重复出现

错误类型 14 unknown出现多次

错误类型 15 welcome出现多次

错误类型 16 exit出现多次

错误类型 17 函数名和关键词冲突

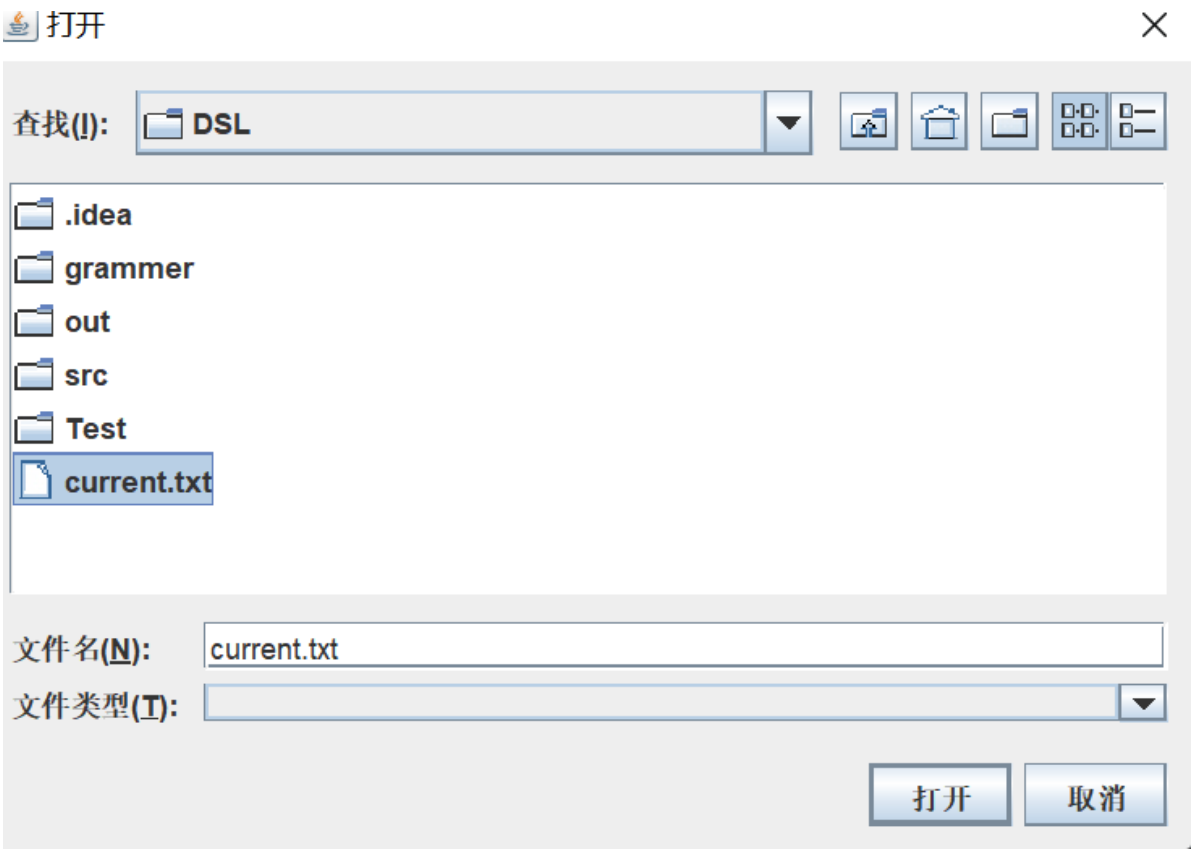
错误类型 18 speak 没有跟语句

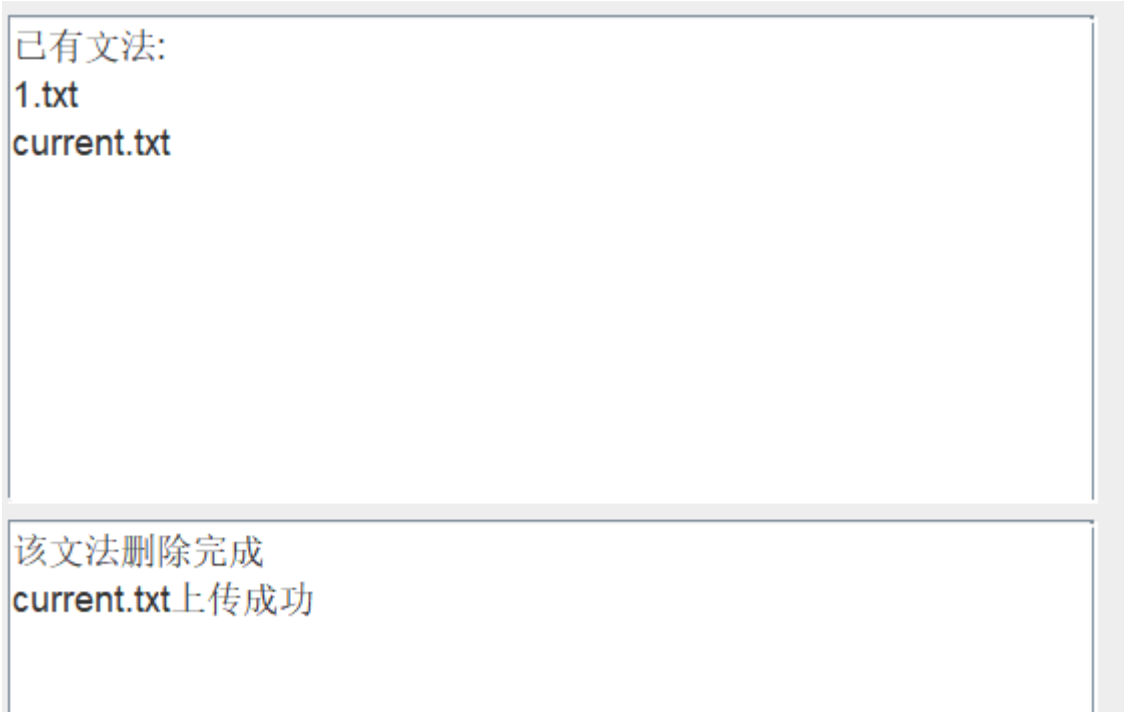
错误类型 19 同一句branch出现过多次

4 使用说明

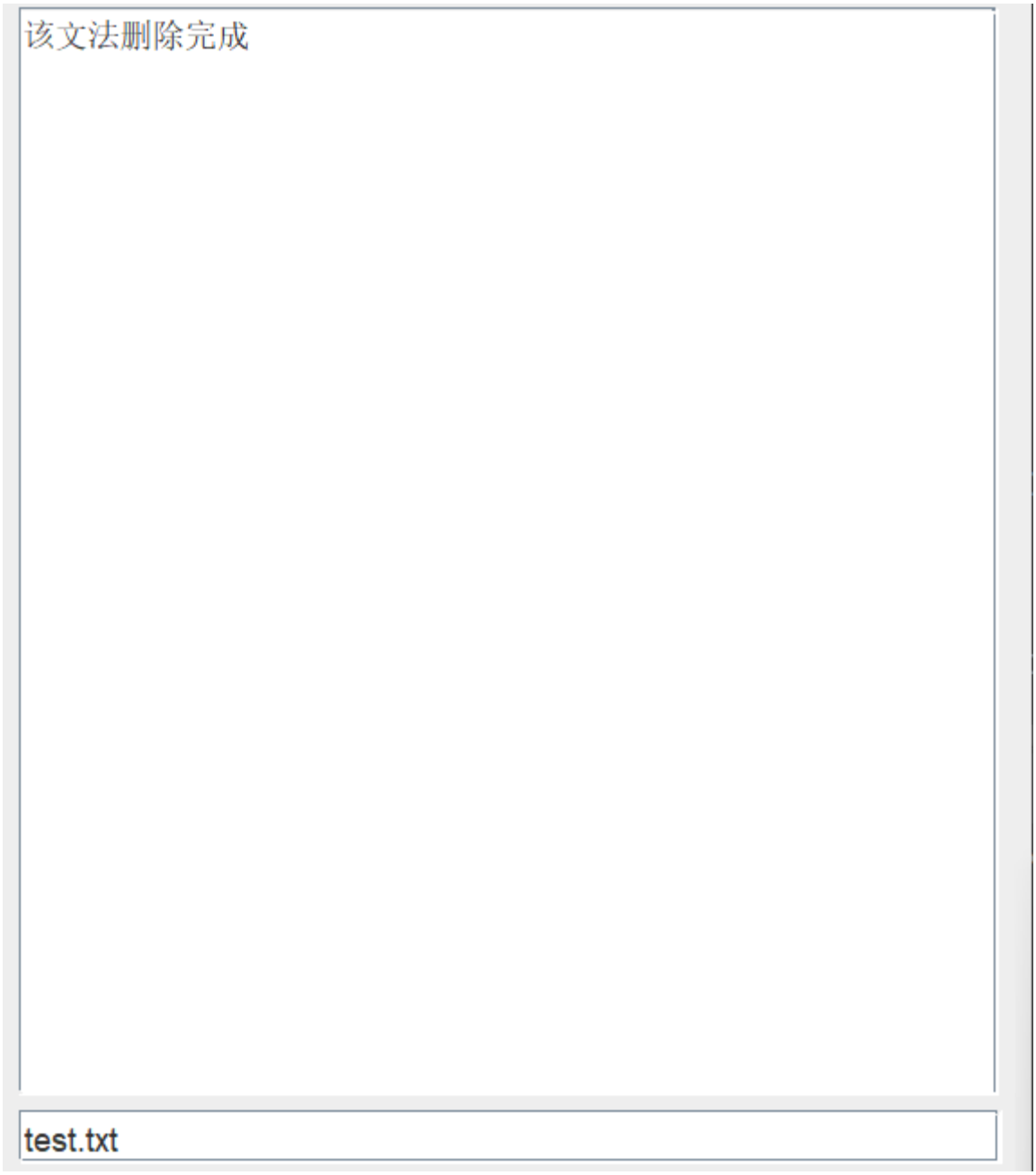
导入文法

1.导入





2.删除



3.选择

如果不满足文法格式要求会报错

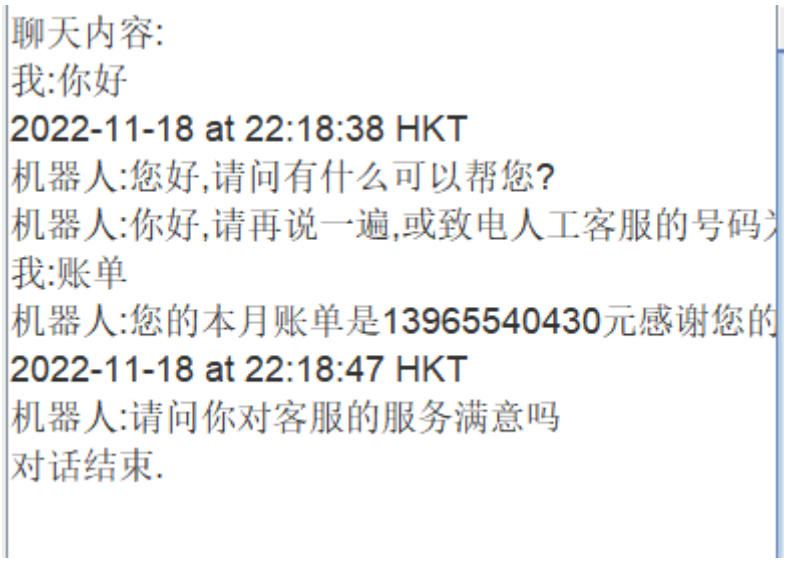
```
1.txt内容为:
Varchar / num /13965540430
        Run/complainProc;Speak /"对不起,问题马上进行解决"
        Run/returnMoneyProc;Speak /"对不起,马上帮你退款";

        Welcome ; Speak/"您好，请问有什么可以帮您?";

        Branch / "投诉" ;Process/ complainProc ;Speak"您的意见是我

        Branch /"账单";Speak / "您的本月账单是"  /num/"元"/"感谢您
       Branch /"退钱" ;Process/ returnMoneyProc ;Speak/"已经退款


        Exit/5;Speak /"请问你对客服的服务满意吗"

        Unknown; Speak/ "你好,请再说一遍,或致电人工客服的号码为


        Remark;顾客是上帝
选择该文法失败!
在第7行出现错误:关键词错误
```

满足格式要求会成功导入

```
1.txt内容为:
Varchar / num /13965540430
        Run/complainProc;Speak /"对不起,问题马上进行解决"
        Run/returnMoneyProc;Speak /"对不起,马上帮你退款";

        Welcome ; Speak/"您好，请问有什么可以帮您?";

        Branch / "投诉" ;Process/ complainProc ;Speak/"您的意见是

        Branch /"账单";Speak / "您的本月账单是"  /num/"元"/"感谢您
       Branch /"退钱" ;Process/ returnMoneyProc ;Speak/"已经退款


        Exit/5;Speak /"请问你对客服的服务满意吗"

        Unknown; Speak/ "你好,请再说一遍,或致电人工客服的号码为


        Remark;顾客是上帝
选择该文法成功!
```

测试

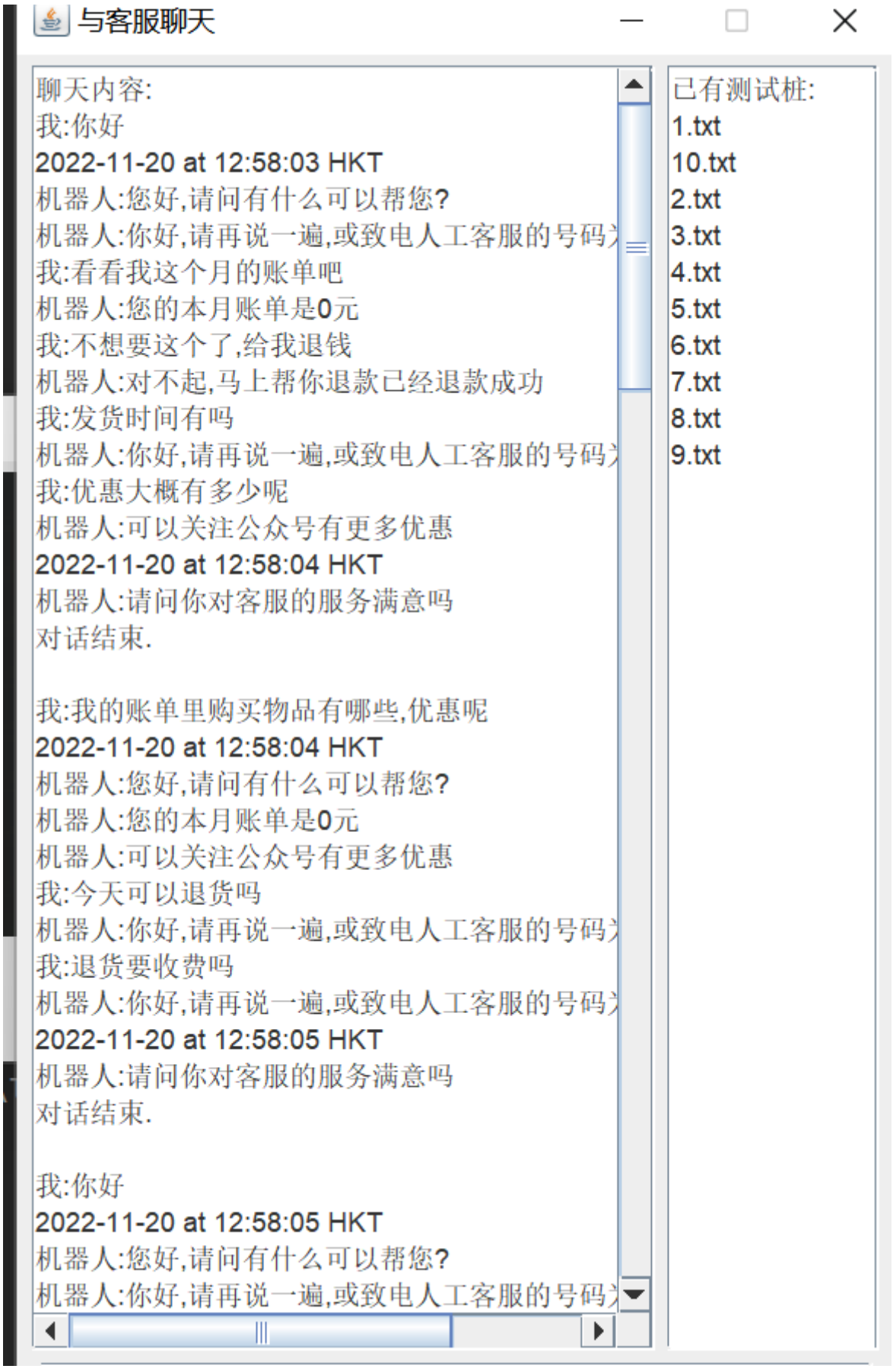
1.对话框模式(文法如上)



2.自动导入测试桩

点击测试键

获得所有测试桩的测试结果



5 实验心得

第一次听过该实验是没有很多头绪,第一是因为没有充分了解编译原理,所以对构建一个文法不是很熟悉,其次第一次如此正式的制作一个软件,对于测试桩接口等概念不熟悉,命名也不是很规范.但随着学习深入,我在编译原理课程上完成了一些实验后充分了解了如何去设置一个文法,并进行编译.同时我也通过前几次程序设计作业了解了命名规范和接口规范,这让我收获很多,也让我更有经验去设计一个较好的程序.

事实上搞明白了文法设计以后这个程序就不难,parser模块花费了我最多的时间,其他模块就不是那么难,因为上学期学java也积累了很多代码,很多ui代码或者读文件的代码可以直接使用,包括Exit控制时间那个模块,我直接套用了上学期学的多线程编程扫雷代码.最后写出程序以后成就感很大.我认为这次程序设计实践给我带来很大的成长.