程序设计实践

姓　　名 李俊达

学　　院 计算机学院

班　　级 2021211304

学　　号 2021210976

目录

问题背景

领域特定语言（Domain Specific Language，DSL）可以提供一种相对简单的文法，用特定领域的业务流程定制。本作业要求定义一个领域特定脚本语言，这个语言能够描述在客服机器人（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。

设计需求

脚本语言的语法可以自由定义，只要语义上满足描述客服机器人自动应答逻辑的要求。
程序输入输出形式不限，可以简化为纯命令行界面。
应该给出几种不同的脚本范例，对不同脚本范例解释器执行之后会有不同的行为表现。

程序介绍：

开发环境：

IntelliJ IDEA 2023.1.3
DataGrip 2023.2
Maven 3.6.3
Tomcat 9.0.27
Mysql 8.0.26
Druid 数据池

采用技术

数据库 sql 语句
后端 java se jdbc
接口 servlet
前端 html css jsp

实现功能

1.登录注册
2.读取脚本
3.解析脚本

4.执行脚本
5.播放语音
6.录制音频
7.查询数据
8.更改余额
9.记录日志
10.增加账单

模块划分

数据库

表名：customers
属性：

Name	用户名	varchar(20)	primary key
Password	密码	varchar(20)	not null
MoneyAmount	存款	int	check (MoneyAmount >0)
Bills	账单	int	check(Bills >0)

ScriptParser

Java 类：
Step　定义 step 结构
数据结构：
StepId（String 类型）：表示步骤的唯一标识符。
Expression（String 类型）：存储播放的语言。
startTimer（int 类型）：表示开始计时器的值。
stopTimer（int 类型）：表示停止计时器的值。
Take（int 类型）：表示取钱的值。
Save（int 类型）：表示存钱的值。
Silence（String 类型）：存储用户不说话跳转的 stepId。
Default（String 类型）：存储如果客户意愿没有相应匹配，应该跳转的 StepId。
Branches（ArrayList 类型）：存储客户的意愿 stepId 的集合。
函数介绍：
toString()方法：
功能描述：重写 toString()方法，用于将 Step 对象转换为字符串形式，方便查看和调试。
Java 类：
Parser
数据结构
Step s Step 类
ArrayList<Step> Steps Step 集合
函数介绍

SelectStep(String StepId)：根据给定的 StepId 在 Steps 列表中查找对应的 Step 对象，如果找到则返回该对象，否则返回 null。
ParserFile(String fileName)：读取指定文件名的脚本文件，并将其内容解析为 Step 对象列表。
ParserLine(String line)：将给定的字符串按空格分割成 token 数组，并将 token 数组传递给 ProcessTokens 方法进行处理。该方法可能会抛出 SQLException 和 IOException 异常。
ProcessStep(String stepName)：创建一个新的 Step 对象，并将其 StepId 设置为给定的 stepName。
ProcessSpeak(String[] token)：处理 Speak 类型的 token，将其 Expression 属性设置为 ProcessExpression 方法的返回值。
ProcessExpression(String[] token)：处理表达式类型的 token，根据 token 的内容生成相应的表达式字符串。。
ProcessListen(String startTimer, String endTimer)：处理 Listen 类型的 token，将其 startTimer 和 stopTimer 属性设置为给定的参数值。
ProcessSilence(String nextStepId)：处理 Silence 类型的 token，将其 Silence 属性设置为给定的 nextStepId。
ProcessDefault(String nextStepId)：处理 Default 类型的 token，将其 Default 属性设置为给定的 nextStepId，并将当前 Step 对象添加到 Steps 列表中。
ProcessBranch(String answer)：处理 Branch 类型的 token，将其 answer 属性添加到当前 Step 对象的 Branches 列表中。
ProcessTake(String TakeAmount)：处理 Take 类型的 token，将其 Take 属性设置为给定的 TakeAmount。
ProcessSave(String SaveAmount)：处理 Save 类型的 token，将其 Save 属性设置为给定的 SaveAmount。
ReturnSteps()：返回 Steps 列表。
ProcessTokens(ArrayList List)：根据给定的 token 列表，调用相应的处理方法进行处理。
模块名称

ScriptInterpreter

Java 类：
Interpreter
函数介绍
executeStep(Step s)：处理一个步骤，包括执行 Speak（调用 SpekingService）、执行 Listen（调用 ListeningService）和执行存钱和取钱操作(调用 DataService)。
executeBranch(Step entryStep)：处理一个分支中的子步骤，包括执行当前步骤和处理 Silence 和 Default 操作。
InterpreterScript(ArrayList steps)：获取脚本语法树，创建执行环境，依次执行每个步骤及其分支。
模块名称

Servlet

Java 类：
LoginServlet：继承自 HttpServlet 的登录 Servlet 类，用于处理用户登录请求。

doPost(HttpServletRequest request, HttpServletResponse response)：处理 POST 请求的方法，用于接收用户登录表单提交的数据，并进行验证。
参数：request - 客户端发送的 HTTP 请求；response - 服务器端发送的 HTTP 响应。
功能描述：从请求中获取用户名和密码，调用 DataService.login()方法进行验证。如果验证成功，调用 LogWrite.Write()方法记录登录日志，并向客户端输出空字符串（调用 JavaScript 的弹窗功能）。如果验证失败，同样调用 LogWrite.Write()方法记录登录日志。
Java 类：
RegisterServlet: 继承自 HttpServlet 的注册 Servlet 类，用于处理用户注册请求。
doPost(HttpServletRequest request, HttpServletResponse response)：处理 POST 请求的方法，用于接收用户注册表单提交的数据，并进行验证和注册操作。
参数：request - 客户端发送的 HTTP 请求；response - 服务器端发送的 HTTP 响应。
功能描述：从请求中获取用户名、密码和确认密码，进行验证和注册操作。如果验证成功，调用 DataService.CreateAccount()方法创建新账户，并将用户重定向到 index.jsp 页面。如果验证失败，调用 LogWrite.Write()方法记录登录日志
模块名称：

Druid

Java 类：
JDBCUtils 封装 JDBC 的工具。
封装方法
public static DataSource getDataSource() 获取连接池对象
public static Connection getConnection() 获取 Connection 对象
public static void close(Connection conn) 关闭连接
模块名称

CustomerData

Java 类：
DataService ：作为数据库与 java 后端的接口
public static String username;：声明一个静态变量 username，用于存储当前登录的用户名。
函数介绍
public static void CreateAccount(String name, String password) throws SQLException：创建一个新用户的方法。接收两个参数，分别是用户名和密码。该方法会将用户信息插入到数据库中的 customers 表中。
public void UpdateMoney(String name, int money) throws SQLException, IOException：更新用户余额的方法。接收两个参数，分别是用户名和要存入或取出的金额。该方法会根据传入的金额进行相应的数据库操作，并记录日志和输出提示信息。
public static boolean isExist(String name) throws SQLException：判断用户是否已存在的方法。接收一个参数，即用户名。该方法会查询数据库中是否存在该用户的记录，并返回结果。
public int SelectMoneyAmount(String name) throws SQLException, IOException：查询用户余额的方法。接收一个参数，即用户名。该方法会查询数据库中该用户的余额，并返回结果。
public int SelectBills(String name) throws SQLException, IOException：查询用户账单的方法。接

收一个参数，即用户名。该方法会查询数据库中该用户的账单，并返回结果。
public static boolean login(String name, String password) throws SQLException：用户登录验证的方法。接收两个参数，分别是用户名和密码。该方法会查询数据库中该用户的密码，并与传入的密码进行比较，如果匹配则返回 true，否则返回 false。
模块名称

DiaryKeeper

Java 类：
LogWrite ：撰写日志
数据结构：
字符串（String）：用于存储用户名和状态信息。
日期时间对象（Calendar）：用于获取当前日期和时间。
日期格式化对象（SimpleDateFormat）：用于将日期时间对象格式化为指定的字符串格式。
缓冲流（BufferedWriter）：用于将数据写入文件。
函数介绍：
Write(String username, String status)：该函数用于将用户名、日期时间和状态信息写入日志文件。
username：表示用户名的字符串参数。
status：表示状态信息的字符串参数。
函数首先定义了日志文件的路径，并创建了一个日期时间对象和一个日期格式化对象。
然后，函数将用户名、格式化后的日期时间和状态信息拼接成一个字符串。
接下来，函数创建了一个缓冲流，并将字符串写入日志文件中。最后关闭缓冲写入器。
模块名称

ListeningSentence

Java 类：
ListeningService :提供录音服务
数据结构
TargetDataLine targetDataLine：用于获取音频输入数据的行。
AudioFormat.Type fileType：定义音频数据的格式，包括采样率、位深度、声道数等。
DataLine.Info info：包含目标数据的信息，如音频格式等。
AudioInputStream audioInputStream：用于读取音频输入流。
File audioFile：表示音频文件的路径和名称。
Timer timer：用于定时执行任务。
TimerTask：表示要执行的任务。
函数介绍
startRecording(int recordTimeInSeconds)：
开始录音，参数为录音时间（秒）。
设置音频格式。获取音频数据行信息。
打开数据行并开始录音。创建音频文件。
创建计时器以停止录音。创建音频输入流并将录音数据写入文件。

stopRecording()：
停止录音。如果数据行不为空且已打开，则停止数据行并关闭它。
Listening(int recordTime)：
启动录音服务，参数为录音时间（秒）。
创建 ListeningService 对象。
打印开始录音的消息。
调用 startRecording 方法开始录音。
打印录音成功的消息。
模块名称

SpeakSentence

Java 类：
TalkService 提供语音服务
数据结构
ActiveXComponent：一个用于与 Windows COM 组件进行交互的类，本例中用于操作 Sapi.SpVoice 和 Sapi.SpFileStream 组件。
Dispatch：一个用于调用 COM 对象方法的类，本例中用于调用 spVoice 和 spFileStream 对象的方法和属性。
Variant：一个用于存储 COM 对象值的类，本例中用于存储 spVoice 和 spFileStream 对象的参数和返回值。
函数介绍
Speaking(String text)：一个用于将给定文本转换为语音并保存为 WAV 文件的方法。
参数：text - 需要转换为语音的文本字符串。
功能描述：该方法首先创建一个 Sapi.SpVoice 对象，设置音量、朗读速度等属性，然后调用其 Speak 方法将文本转换为语音。接着，创建一个 Sapi.SpFileStream 对象，设置音频流格式，并将其设置为 spVoice 对象的音频输出流。最后，将生成的语音保存为 WAV 文件，并关闭输出文件。
前端

Index.jsp

功能模块：
用户登录表单：用户可以输入用户名和密码，点击登录按钮提交表单
登录检测：接收前端发送的登录请求，验证用户名和密码是否正确，返回相应的登录结果
注册链接：用户可以点击链接跳转到注册页面进行注册
代码实现：
HTML 部分：包含登录表单和注册链接
CSS 部分：定义页面样式和布局
JavaScript 部分：处理表单提交事件和验证逻辑
Java 部分：在 LoginServlet 类中，处理登录请求和验证用户信息

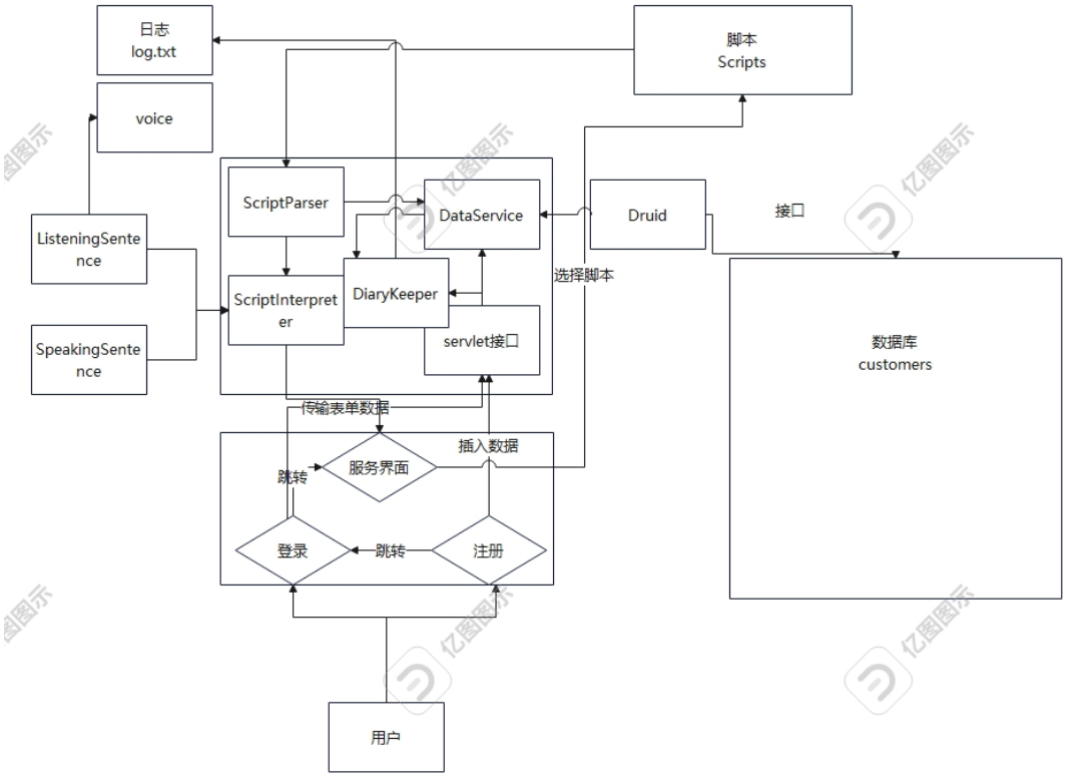
Register.jsp

功能模块：
用户注册表单：用户可以输入用户名，密码，确认密码，点击登录按钮提交表单
注册检测：接收前端发送的注册请求，验证用户名是否为空或者已存在和密码与确认密码是否一致，返回相应的登录结果
代码实现：
HTML 部分：包含注册表单
CSS 部分：定义页面样式和布局
JavaScript 部分：处理表单提交事件和验证逻辑
Java 部分：实现注册检测和向数据库中插入数据

RobotService.jsp

功能模块：
模式选择表单：用户输入编号，选择脚本。
启动服务按钮：点击按钮启动服务
提供客服服务：
代码实现：
HTML 部分：包含模式选择表单和启动服务按钮。
CSS 部分：定义页面样式和布局。
Java 部分：调用后端代码，提供客服服务。

模块关系



接口实现

人机接口

登录界面

填写用户名和密码，如果没有账号点击注册链接跳转注册界面。
注册界面

填写用户名，密码和确认密码，密码，确认密码一致时注册成功跳转登录界面。

填写脚本编号，选择执行脚本，点击启动服务按钮，启动客服服务

程序接口

元素 1	接口	元素 2
DataService	JDBCUtils	数据库
Parser	ReturnSteps()	Interpreter
Index.jsp	LoginServlet	DataService
Register.jsp	RegisterServlet	DataService
TalkService	Speak()	Interpreter
ListenService	Listen()	Interpreter
Interpreter	InterpreterScript	RebotService.jsp
LogWrite	Write()	DataService LoginServlet RegisterServlet

测试脚本

脚本 1

```
Step welcome
Speak $name + "你好，我能为你做什么?"
Listen 5, 20
Branch "投诉", complainProc
Branch "账单", billProc
Silence silenceProc
Default defaultProc
Step 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?"
Listen 5, 50
Default thanks
Step thanks
Speak "感谢您的来电，再见"
Exit
Step billProc
```

```
Speak "您的本月账单是" + $amount + "元，感谢您的来电，再见"
Exit
Step silenceProc
Speak "听不清，请您大声一点可以吗"
Listen 5, 20
Branch "投诉", complainProc
Branch "账单", billProc
Silence silenceProc
Default defaultProc
Step defaultProc
Speak "谢谢惠顾"
Exit
```

脚本 2

```
Step welcome
Speak $name + "你好，我能为你做什么?"
Listen 5, 25
Branch "余额", restProc
Branch "账单", billProc
Silence silenceProc
Default defaultProc
Step restProc
Speak "您的余额是" + $rest + "元，感谢您的来电，再见"
Listen 5, 45
Default thanks
Step thanks
Speak "感谢您的来电，再见"
Exit
Step billProc
Speak "您的本月账单是" + $amount + "元，感谢您的来电，再见"
Exit


Step silenceProc
Speak "听不清，请您大声一点可以吗"
Listen 5, 25
Branch "余额", restProc
Branch "账单", billProc
Silence silenceProc
Default defaultProc
Step defaultProc
Speak "谢谢惠顾"
Exit
```

脚本 3

```
Step welcome
Speak $name + "你好，我能为你做什么?"
Listen 5, 20
Branch "存钱", saveProc
Branch "取钱", takeProc
Branch "余额", restProc
Default defaultProc
Step saveProc
Speak "您的账户已从绑定的微信钱包中提出并存入 1000 元，感谢您的来电，再见"
Save 1000
Exit
Step takeProc
Speak "您的账户已提现 1000 元，收取 1 元手续费，感谢您的来电，再见"
Take 1000
Exit
Step restProc
Speak "您的余额是" + $rest + "元，感谢您的来电，再见"
Default thanks
Step thanks
Speak "感谢您的来电，再见"
Exit
Default defaultProc
Step defaultProc
Speak "谢谢惠顾"
Exit
```

脚本 4

```
Step welcome
Speak $name + "你好，我能为你做什么?"
Listen 5, 20
Branch "存钱", saveProc
Branch "取钱", takeProc
Silence silenceProc
Default defaultProc
Step saveProc
Speak "您的账户已从绑定的微信钱包中提出并存入 1000 元，感谢您的来电，再见"
Save 1000
Silence askProc
Default thanks
Step askProc
Speak "存钱成功，您还有其他需求吗？"
Listen 5, 20
Default answers
# 把 Step 写成 step 发生语法错误
```

```
Step answers
Speak "收到您的要求，已开始执行"
Exit
Step thanks
Speak "感谢您的来电，再见"
Exit
Step takeProc
Speak "您的账户已提现 1000 元，收取 1 元手续费，感谢您的来电，再见"
Take 1000
Exit
Step silenceProc
Speak "听不清，请您大声一点可以吗？"
Listen 5, 20
Branch "存钱", saveProc
Branch "取钱", takeProc
Silence silenceProc
Default defaultProc
Step defaultProc
Speak "谢谢惠顾"
Exit
```

脚本 5

```
Step welcome
Speak $name + "你好，我能为你做什么?"
Listen 5, 20
Branch "存钱", saveProc
Branch "投诉", complainProc
Silence silenceProc
Default defaultProc
Step saveProc
Speak $name + "您的账户已从绑定的微信钱包中提出并存入 3000 元，感谢您的来电，再见"
Save 3000
Default finish
Step finish
Speak "服务完成"
Exit
Step complainProc
Speak "对不起，请问服务哪里有不周到的地方，欢迎您提出意见！"
Silence implore
Default thanks
Step implore
Speak "请您一定提出意见"
Listen 5, 25
Exit
```

```
Step thanks
Speak "感谢您提出的意见，我们会改进的"
Exit
Step silenceProc
Speak "听不清，请您大声一点可以吗"
Listen 5, 20
Branch "存钱", saveProc
Branch "投诉", complainProc
Silence silenceProc
Default defaultProc
Step defaultProc
Speak "谢谢惠顾"
Exit
```

自动测试

测试类
TestSpeaking
测试方法
Speaking(): 测试了 TalkService 类的 speak 方法，播放语音
测试类
TestListening
测试方法
Listening(): 测试了 ListeningService 类的 Listening 方法，启动电脑麦克风开启录音。
测试类
TestDataService
测试方法
TestUpdate(): 测试了 DataService 类的 UpdateMoney 方法，用于更新用户的余额。它创建了一个 DataService 对象，并调用 UpdateMoney 方法来更新用户"李俊达"的余额为 100。
TestLogin(): 测试了 DataService 类的 login 方法，用于验证用户的登录信息。如果登录成功，它会调用 TalkService 类的 Speaking 方法来播放一条消息"登录成功"，并在控制台输出"登录成功"。否则，它会在控制台输出"登录失败"。
TestMoneyTake(): 测试了 DataService 类的 UpdateMoney 方法，用于从用户的余额中扣除一定的金额。它创建了一个 DataService 对象，并调用 UpdateMoney 方法来从用户"李俊达"的余额中扣除 40000。
TestSelectMoney(): 测试了 DataService 类的 SelectMoneyAmount 方法，用于查询用户的余额。它创建了一个 DataService 对象，并调用 SelectMoneyAmount 方法来获取用户"李俊达"的余额，然后在控制台输出该余额。
TestSelectBills(): 测试了 DataService 类的 SelectBills 方法，用于查询用户的账单。它创建了一个 DataService 对象，并调用 SelectBills 方法来获取用户"李俊达"的账单，然后在控制台输出该账单。
测试类

TestParser
测试方法
TestSteps():输出脚本解析的 step 集合的字符串形式
测试结果
S1:
Step{StepId='welcome', Expression='你好，我能为你做什么?', startTimer=5, stopTimer=20, Take=0, Save=0, Silence='silenceProc', Default='defaultProc', Branches=[complainProc, billProc]}
Step{StepId='complainProc', Expression='您的意见是我们改进工作的动力，请问您还有什么补充?', startTimer=5, stopTimer=50, Take=0, Save=0, Silence='null', Default='thanks', Branches=[]}
Step{StepId='thanks', Expression='感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='billProc', Expression='您的本月账单是 0 元，感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='silenceProc', Expression='听不清，请您大声一点可以吗', startTimer=5, stopTimer=20,Take=0,Save=0,Silence='silenceProc',Default='defaultProc',Branches=[complainProc, billProc]}
Step{StepId='defaultProc', Expression='谢谢惠顾', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
S2：
Step{StepId='welcome', Expression='你好，我能为你做什么?', startTimer=5, stopTimer=25, Take=0, Save=0, Silence='silenceProc', Default='defaultProc', Branches=[restProc, billProc]}
Step{StepId='restProc', Expression='您的余额是 0 元，感谢您的来电，再见', startTimer=5, stopTimer=45, Take=0, Save=0, Silence='null', Default='thanks', Branches=[]}
Step{StepId='thanks', Expression='感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='billProc', Expression='您的本月账单是 0 元，感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='silenceProc', Expression='听不清，请您大声一点可以吗', startTimer=5, stopTimer=25, Take=0, Save=0, Silence='silenceProc', Default='defaultProc', Branches=[restProc, billProc]}
Step{StepId='defaultProc', Expression='谢谢惠顾', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
S3：
Step{StepId='welcome', Expression='你好，我能为你做什么?', startTimer=5, stopTimer=20, Take=0, Save=0, Silence='null', Default='defaultProc', Branches=[saveProc, takeProc, restProc]}
Step{StepId='saveProc', Expression='您的账户已从绑定的微信钱包中提出并存入 1000 元，感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0, Save=1000, Silence='null', Default='null', Branches=[]}
Step{StepId='takeProc', Expression='您的账户已提现 1000 元，收取 1 元手续费，感谢您的来电，再见', startTimer=0, stopTimer=0, Take=1000, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='restProc', Expression='您的余额是 0 元，感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='thanks', Branches=[]}
Step{StepId='thanks', Expression='感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0,

Save=0, Silence='null', Default='defaultProc', Branches=[]}
Step{StepId='defaultProc', Expression='谢谢惠顾', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
S4：
Step{StepId='welcome', Expression='你好，我能为你做什么?', startTimer=5, stopTimer=20, Take=0, Save=0, Silence='silenceProc', Default='defaultProc', Branches=[saveProc, takeProc]}
Step{StepId='saveProc', Expression='您的账户已从绑定的微信钱包中提出并存入 1000 元，感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0, Save=1000, Silence='askProc', Default='thanks', Branches=[]}
Step{StepId='askProc', Expression='存钱成功，您还有其他需求吗？', startTimer=5, stopTimer=20, Take=0, Save=0, Silence='null', Default='answers', Branches=[]}
Step{StepId='answers', Expression='收到您的要求，已开始执行', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='thanks', Expression='感谢您的来电，再见', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='takeProc', Expression='您的账户已提现 1000 元，收取 1 元手续费，感谢您的来电，再见', startTimer=0, stopTimer=0, Take=1000, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='silenceProc', Expression='听不清，请您大声一点可以吗？', startTimer=5, stopTimer=20, Take=0, Save=0, Silence='silenceProc', Default='defaultProc', Branches=[saveProc, takeProc]}
Step{StepId='defaultProc', Expression='谢谢惠顾', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
S5：
Step{StepId='welcome', Expression='你好，我能为你做什么?', startTimer=5, stopTimer=20, Take=0, Save=0, Silence='silenceProc', Default='defaultProc', Branches=[saveProc, complainProc]}
Step{StepId='saveProc', Expression='您的账户已从绑定的微信钱包中提出并存入 3000 元，感谢您的来电,再见', startTimer=0, stopTimer=0, Take=0, Save=3000, Silence='null', Default='finish', Branches=[]}
Step{StepId='finish', Expression='服务完成', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='complainProc', Expression='对不起，请问服务哪里有不周到的地方，欢迎您提出意见！', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='implore', Default='thanks', Branches=[]}
Step{StepId='implore', Expression='请您一定提出意见', startTimer=5, stopTimer=25, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='thanks', Expression='感谢您提出的意见，我们会改进的', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
Step{StepId='silenceProc', Expression='听不清，请您大声一点可以吗', startTimer=5, stopTimer=20, Take=0, Save=0, Silence='silenceProc', Default='defaultProc', Branches=[saveProc, complainProc]}
Step{StepId='defaultProc', Expression='谢谢惠顾', startTimer=0, stopTimer=0, Take=0, Save=0, Silence='null', Default='null', Branches=[]}
测试类

TestInterpreter
测试方法
PrintProcess()输出解释结果
测试结果
S1:
你好，我能为你做什么?
开始录音
录音成功
录音 15s
录音结束
您的意见是我们改进工作的动力，请问您还有什么补充?
开始录音
录音成功
录音 45s
录音结束
感谢您的来电，再见
您的本月账单是 0 元，感谢您的来电，再见
听不清，请您大声一点可以吗
开始录音
录音成功
录音 15s
录音结束
谢谢惠顾
Success Interpreter
S2
你好，我能为你做什么?
开始录音
录音成功
录音 20s
录音结束
您的余额是 0 元，感谢您的来电，再见
开始录音
录音成功
录音 40s
录音结束
感谢您的来电，再见
您的本月账单是 0 元，感谢您的来电，再见
听不清，请您大声一点可以吗
开始录音
录音成功
录音 20s
录音结束
谢谢惠顾
Success Interpreter

S3
你好，我能为你做什么?
开始录音
录音成功
录音 15s
录音结束
您的账户已从绑定的微信钱包中提出并存入 1000 元，感谢您的来电，再见
null 在河南村镇银行存了 1000 元
您的账户已提现 1000 元，收取 1 元手续费，感谢您的来电，再见
余额不足
您的余额是 0 元，感谢您的来电，再见
感谢您的来电，再见
谢谢惠顾
Success Interpreter
S4
你好，我能为你做什么?
开始录音
录音成功
录音 15s
录音结束
您的账户已从绑定的微信钱包中提出并存入 1000 元，感谢您的来电，再见
null 在河南村镇银行存了 1000 元
存钱成功，您还有其他需求吗？
开始录音
录音成功
录音 15s
录音结束
感谢您的来电，再见
您的账户已提现 1000 元，收取 1 元手续费，感谢您的来电，再见
听不清，请您大声一点可以吗？
开始录音
录音成功
录音 15s
录音结束
谢谢惠顾
Success Interpreter
S5
你好，我能为你做什么?
开始录音
录音结束
录音成功
录音 15s
您的账户已从绑定的微信钱包中提出并存入 3000 元，感谢您的来电，再见
null 在河南村镇银行存了 3000 元

服务完成
对不起，请问服务哪里有不周到的地方，欢迎您提出意见！
请您一定提出意见
开始录音
录音成功
录音 20s
录音结束
感谢您提出的意见，我们会改进的
听不清，请您大声一点可以吗
开始录音
录音成功
录音 15s
录音结束
Success Interpreter

脚本语法

关键词

Step: 完整表示一个步骤的所有行为
Speak:
计算表达式合成一段文字
调用媒体服务器进行语音合成并播放
Listen:
调用媒体服务器对客户说的话录音，并进行语音识别
语音识别的结果调用“自然语言分析服务”分析客户的意愿
Take: 调用数据服务，减少客户余额，记录在客户账单
Save: 调用数据服务，增加客户余额。
Branch:
对客户的意愿进行分支处理，不同的意愿，跳转到不同的 Step
Silence: 如果用户不说话，应该跳转到哪个 Step
Default: 如果客户意愿没有相应匹配，应该跳转到哪个 Step
Exit: 结束对话

变量

Name:客户姓名
Rest:客户余额
Bill：客户账单

注释

使用井号 # 作为单行注释的符号，语法格式为： #注释内容

写法

声明 Step:
Step StepId
StepId 为 Step 的名字
示例

```
Step welcome
```

调用 Speak
Speak “text”
text 为中文语句，用英文双引号””。
示例

```
Speak "您的意见是我们改进工作的动力，请问您还有什么补充?"
```

text:
调用变量前用$
语句间用+号连接
示例

```
"您的本月账单是" + $amount + "元，感谢您的来电，再见"
```

调用 Listen
Listen starttime, endtime
Starttime aendtime 均为正整数，规定录音起止时间，用英文,分隔。
示例

```
Listen 5, 50
```

调用 Branch
Branch "comment", StepId
Comment 是注释，解释 branch 的目的,StepId 是客户意愿的 Step
Branch 只在脚本第一个 step 中才能调用和声明执行。
示例

```
Branch "投诉", complainProc
```

Silence StepId
声明客户处于沉默时执行的 StepId
示例

```
Silence silenceProc
```

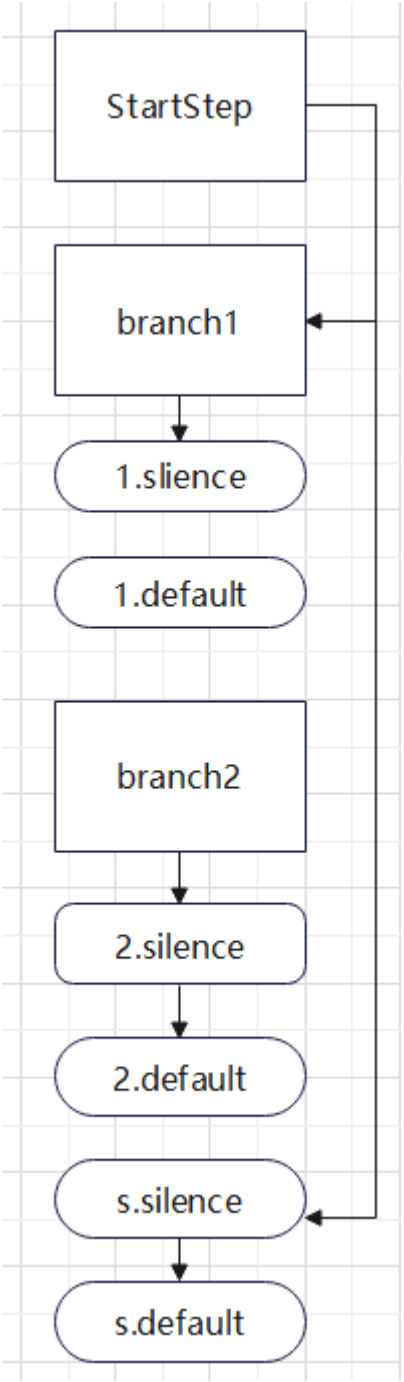
Default StepId
声明客户意愿没有相应匹配时执行的 StepId
示例

```
Default thanks
```

结构

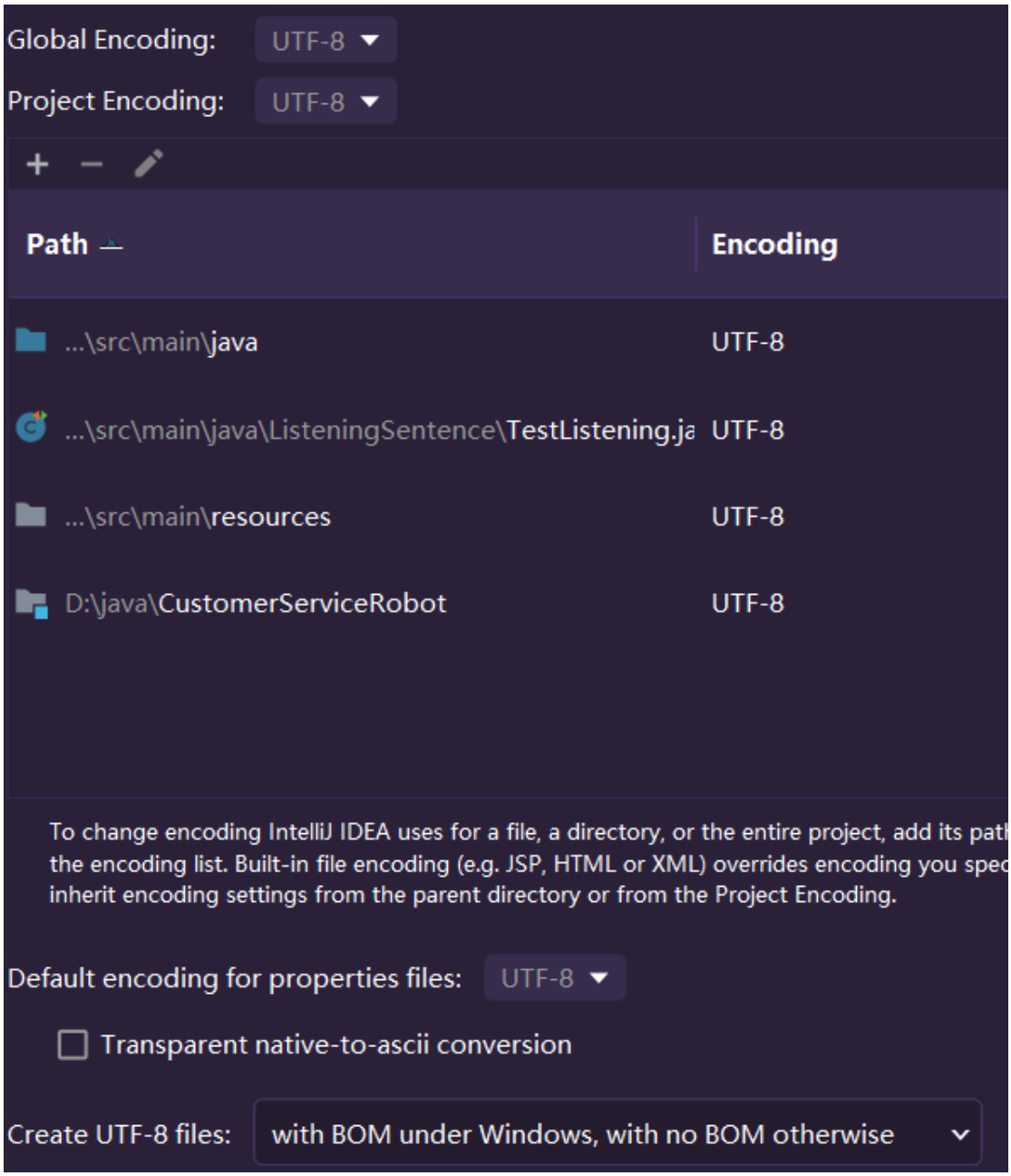
脚本声明的第一个 Step 是整个脚本的主程序，Branch 内声明子程序。
然后依次写子 step，如果子 step 内存在 silence 和 default，在该子 step 后依次写 silence step 和 default step，最后如果主程序存在 silence defualt

依次写主程序的 silence step 和 default step，书写顺序
如下图所示。



调试日志：

程序中文显示乱码



均设为 utf-8。问题解决

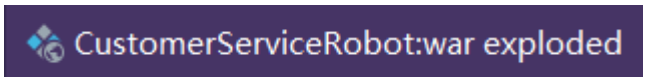
用户名为空时仍能成功注册

```
if(username.length()==0 || password.length()==0 || ConfrimPassword.length()==0){
    out.print("<script>alert('信息不完整'); window.location='Register.jsp' </script>");
    LogWrite.Write(username, status: "注册失败");
    out.flush();
    out.close();
}
```

增加信息检测功能，成功解决。
@Test 注解失效
启动 maven clean ,maven install 成功解决。
处理取钱数大于用户余额的情况
在函数中执行两个 sql 语句，禁用自动提交

```
LogWrite.Write(name,  status: "取钱失败，余额不足"
TalkService.Speaking( text: "余额不足");
System.out.println("余额不足");
```

代码更新后界面没有更新
Deployment 选择错误



服务界面执行 jsp 内部的 java 代码后才显示网页
设计按钮，点击按钮后运行 java 方法，成功解决