**南京大学软件学院**

12

**详细设计文档**

**软件设计交流系统 第四循环**

**萌军赶死队：刘羽佳 苗羿 卿荣山 王恬 完颜章伟**

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 作者 | 时间 | 内容 |
| 1 | 王恬 | 2012/5/28 | 确定文档范围 |
| 2 | 王恬 | 2012/5/28 | 完善部分内容 |
| 3 | 王恬 | 2012/5/29 | 继续完善 |
| 4 | 王恬 | 2012/5/30 | 继续完善 |
| 5 | 苗羿 | 2012/6/1 | 添加音频聊天部分 |
| 6 | 卿荣山 | 2012/6/1 | 添加电子白板部分 |
| 7 | 完颜章伟 | 2012/6/2 | 添加日志记录部分 |
| 8 | 卿荣山 | 2012/6/2 | 完善电子白板部分 |
| 9 | 王恬 | 2012/6/3 | 添加各模块设计图 |

目录

[1. 引言 4](#_Toc326431631)

[1.1. 编写目的 4](#_Toc326431632)

[1.2. 对象和范围 4](#_Toc326431633)

[1.3. 术语表 4](#_Toc326431634)

[1.4. 参考资料 5](#_Toc326431635)

[1.5. 使用的文字处理和绘图工具 5](#_Toc326431636)

[2. 全局数据结构说明 6](#_Toc326431637)

[2.1. 常量 6](#_Toc326431638)

[2.2. 变量 6](#_Toc326431639)

[2.3. 数据结构 6](#_Toc326431640)

[3. 模块设计 7](#_Toc326431641)

[3.1. 用例图 7](#_Toc326431642)

[3.2. 功能设计说明 7](#_Toc326431643)

[3.2.1. 客户端 7](#_Toc326431644)

[3.2.2. 服务器 20](#_Toc326431645)

[4. 接口设计 27](#_Toc326431646)

[4.1. 内部接口 27](#_Toc326431647)

[4.1.1. 接口说明 27](#_Toc326431648)

[5. 数据库设计 42](#_Toc326431649)

[6. 系统安全保密设计 42](#_Toc326431650)

[6.1. 说明 42](#_Toc326431651)

[6.2. 设计 42](#_Toc326431652)

[6.2.1. 身份验证部分 42](#_Toc326431653)

[6.2.2. 音频聊天部分 43](#_Toc326431654)

[7. 系统性能设计 43](#_Toc326431655)

[8. 系统出错处理 43](#_Toc326431656)

# 引言

## 编写目的

本详细设计说明书的编写目的是为了说明程序系统的各个层次中的每个软件对象（包括每个模块和程序）的设计考虑，以向系统实现（编码和测试）阶段提供关于程序系统实现方式的详细描述，为软件编程和系统维护提供基础。

## 对象和范围

* 文档预期阅读对象：系统设计人员、软件开发人员、软件测试人员、项目评审人员
* 文档范围：软件设计交流系统的模块划分以及模块内部的详细设计

## 术语表

|  |  |  |
| --- | --- | --- |
| 序号 | 术语或缩略语 | 定义 |
| 1 | MySQL | MySQL是一个开放源码的关联式数据库管理系统，在过去由于性能高、成本低、可靠性好，已经成为最流行的开源数据库，因此被广泛地应用在Internet上的中小型网站中。随着MySQL的不断成熟，它也逐渐用于更多大规模网站和应用，比如维基百科、Google和Facebook等网站。非常流行的开源软件组合LAMP中的“M”指的就是MySQL。 |
| 2 | JDBC | Java 数据库连接，简称JDBC，是Java语言中用来规范客户端程序如何来访问数据库的应用程序接口，提供了诸如查询和更新数据库中数据的方法。它是面向关系型数据库的。 |
| 3 | JMF | [Java](http://baike.baidu.com/view/29.htm)术语，意为Java媒体框架（JMF）。该核心框架支持不同[媒体](http://baike.baidu.com/view/7072.htm)（如：音频输出和[视频输出](http://baike.baidu.com/view/168336.htm)）间的时钟同步。它是一个标准的扩展框架，允许用户制作纯[音频流](http://baike.baidu.com/view/1141389.htm)和[视频流](http://baike.baidu.com/view/2722588.htm)。 |

## 参考资料

|  |  |
| --- | --- |
| 名称 | 来源 |
| 《软件详细设计文档模板》 | 百度知道 |
| 《体系架构原型文档》 | 本项目第三循环 |
| 《软件架构设计文档》 | 本项目第三循环 |
| 《计算与软件工程Ⅲprint》 | 教材 |

## 使用的文字处理和绘图工具

|  |  |
| --- | --- |
| 工具名 | 简介 |
| Enterprise Architect | 以目标为导向的软件系统。它覆盖了系统开发的整个周期，除了开发类模型之外，还包括事务进程分析，使用案例需求，动态模型，组件和布局，系统管理，非功能需求，用户界面设计，测试和维护等。 |
| Microsoft Word | Microsoft Word是微软公司的一个文字处理器应用程序。 |

# 全局数据结构说明

由于本软件遵守面向对象设计原则，所以以下各项均为无。

## 常量

* UserTable

## 变量

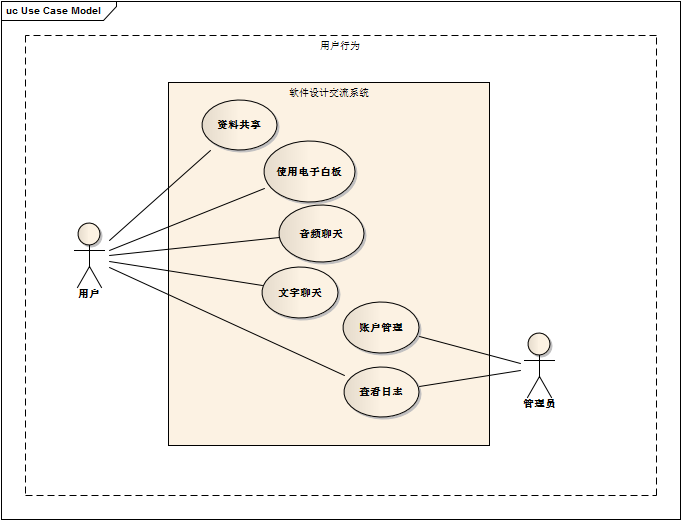
* ClientInfoManager
* DBConnect

## 数据结构

无

# 模块设计

## 用例图



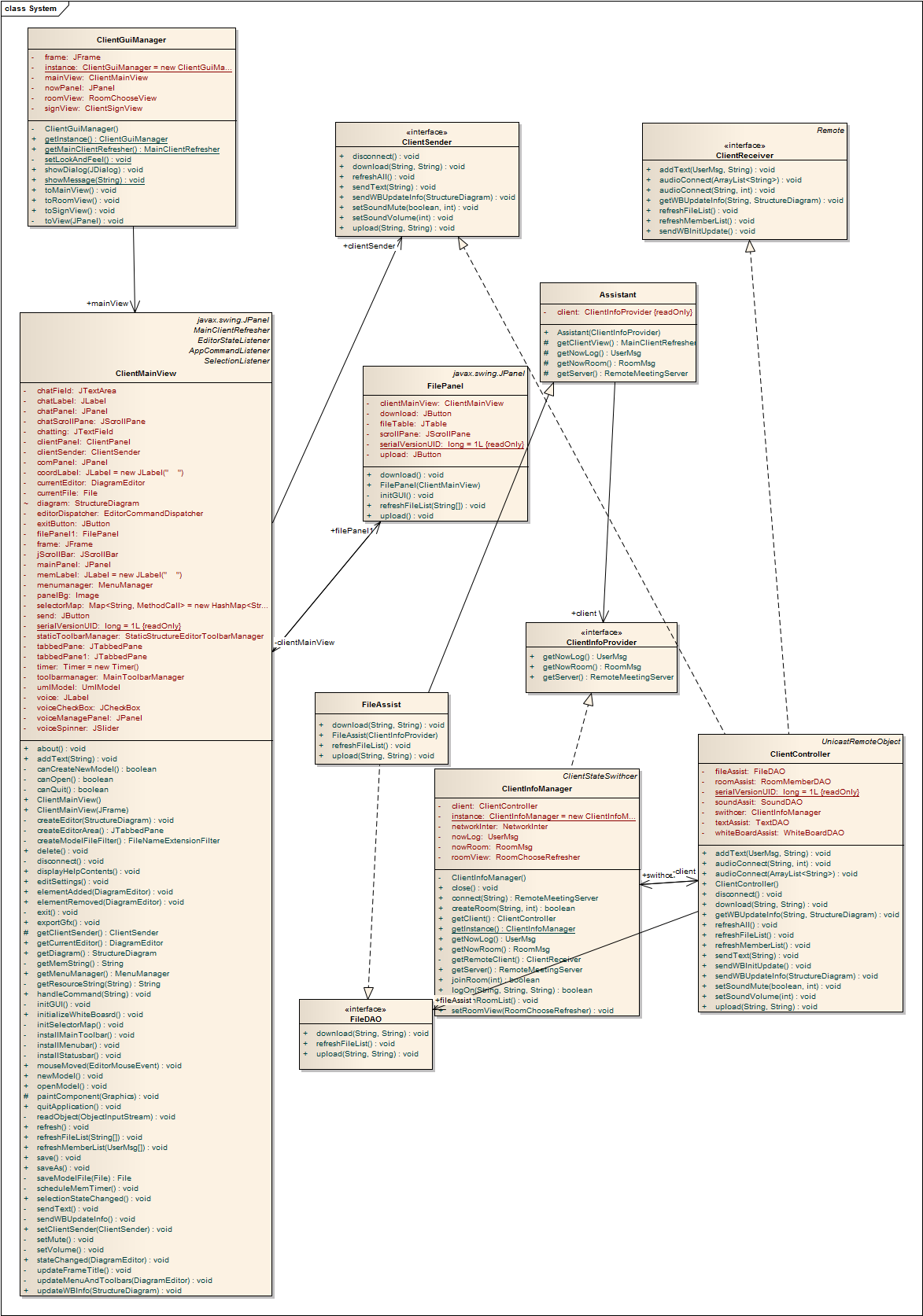
## 功能设计说明

### 客户端

客户端负责发送用户请求和接收处理其他用户的请求，整体实现为MVC架构，分为view，controller，model三层：View主要用于呈现信息，Model用于本地业务逻辑并和数据层交互。Controller则是前两者间的桥梁。下面以功能为模块进行划分并给出详细设计说明。

#### 资料共享

##### 设计图



##### 功能描述

用户可以将本地的文件共享在系统中，供所有用户下载；也可以自由下载其他用户已经共享的文件资料。

##### 输入数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 有效性检验规则 |
| 文件绝对路径 | 要上传的文件的绝对路径，包括文件名 | 不能为空，否则提示未选择文件 |
| 文件绝对路径 | 希望下载文件存放到的文件夹绝对路径，包括自定义的文件名 | 不能为空，否则提示未选择下载到的文件夹 |

##### 输出数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 表现形式 |
| 是否上传成功 | 提示用户上传是否成功 | 弹出窗口 |
| 是否下载成功 | 提示用户下载是否成功 | 弹出窗口 |
| 文件列表 | 上传文件后服务器所有的新文件列表 | 列表刷新 |
| 文件 | 从服务器下载的文件 | 在之前选择的文件夹出现选择下载的文件 |

##### 业务算法和流程

上传文件：客户端根据要上传的文件名将本地文件解析成二进制流，加以文件的名称等信息发送给服务器，服务器接收文件并将文件保存在固定目录，客户端上传成功后提示成功并刷新目录。

下载文件：服务器根据客户端选择下载的文件名将文件解析成二进制流，加以文件的名称等信息发送给客户端，客户端接收文件并将文件保存在之前选定的目录，客户端接收文件成功后进行提示。

##### 数据设计

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据说明 | 数据信息 |
| filePath | 上传文件的绝对路径 | String filePath |
| putPath | 下载文件的绝对路径 | String putPath |
| fileName | 上传/下载文件的名称 | String fileName |
| content | 被解析成二进制流的文件 | byte content[] |

##### 源程序文件说明

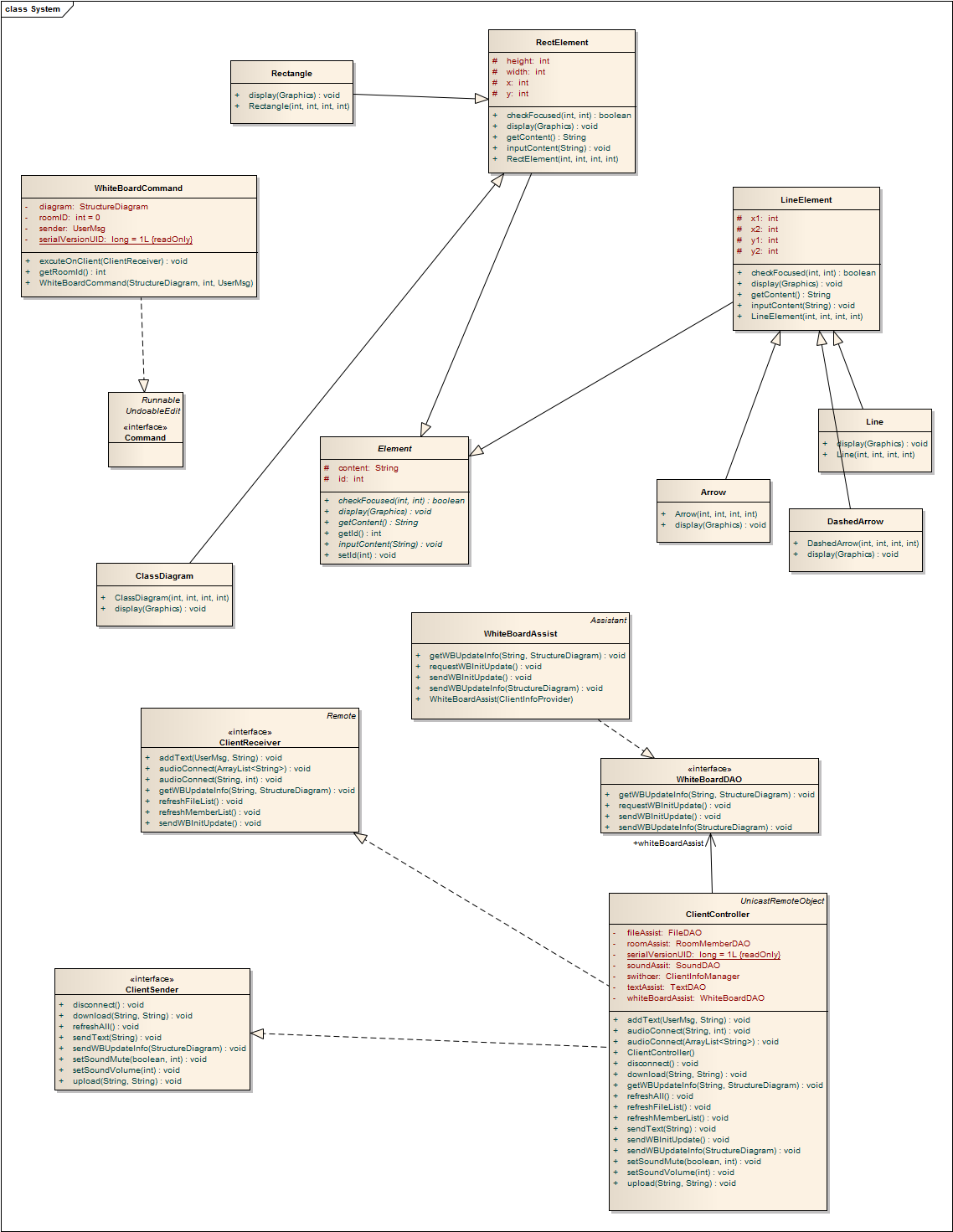
|  |  |  |  |
| --- | --- | --- | --- |
| 文件名称 | 所在目录 | 功能说明 | 重要函数 |
| ClientController | module.client | 负责客户端逻辑控制消息转发 | public void upload(String filePath, String fileName)  public void download(String putPath, String fileName) |
| FileAssist | Control.client | 文件传输核心类 | public void upload(String filePath, String fileName)  public void download(String putPath, String fileName) |
| ClientMainView | View.client | 文件列表显示 | public void refresh() |

##### 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名称 | 参数 | 返回值 | 所在文件 | 算法及功能简介 |
| upload | String filePath,  String fileName | Void | FileAssist | 根据文件路径将文件解析成二进制流 |
| download | String putPath  String fileName | Void | FileAssist | 将二进制流转化成文件存储在本地 |
| refreshFileList | None | Void | ClientController | 在本地刷新文件列表 |
| upload | String filePath,  String fileName | Void | ClientSender | 调用底层实现 |
| download | String putPath  String fileName | void | ClientSender | 调用底层实现 |

#### 电子白板

##### 设计图



##### 功能描述

处于同一房间的在线用户可以通过共享白板讨论软件设计，通过增、删、改基本的UML图元，方便、及时地进行交流。

##### 输入数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 有效性检验规则 |
| 新建图元类别 | 在一个菜单栏里选择所需要新建的元素或者联系 | 点击之后鼠标移至白板处会出现一个方框，否则不出现 |
| 新建图元位置 | 点击选择新建的图元后点击白板上的任意一点选择新建位置 | 一个默认图元被放置在白板上 |
| 图元大小调整 | 单击选中图元，拖动四角来调整大小 | 图元大小发生变动 |

##### 输出数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 表现形式 |
| 白板详细内容 | 包括所有图元（名称、属性）的展示 | 在白板上画出所有图元 |
| 当前鼠标位置 | 鼠标在白板上的位置 | 在下方灰色区域内以坐标形式输出 |

##### 业务算法和流程

当一个用户选择添加、移除或者修改一个白板上的图元的时候，该客户端会将序列化后白板信息发送至服务器，服务器再将该信息转发至该用户所在房间里除了该用户以外的所有用户客户端，然后各客户端根据所接受的修改信息进行修改。以此实现白板的网络交流。

##### 数据设计

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据说明 | 数据信息 |
| mainElements | UML模型中所有元素 | HashSet<UmlModelElement> mainElements |
| connections | UML模型中所有关系 | ArrayList<Connection> connections |
| umlModel | 一个UML模型 | UmlModel umlModel |
| diagram | UML模型的结构 | StructureDiagram diagram |

##### 源程序文件说明

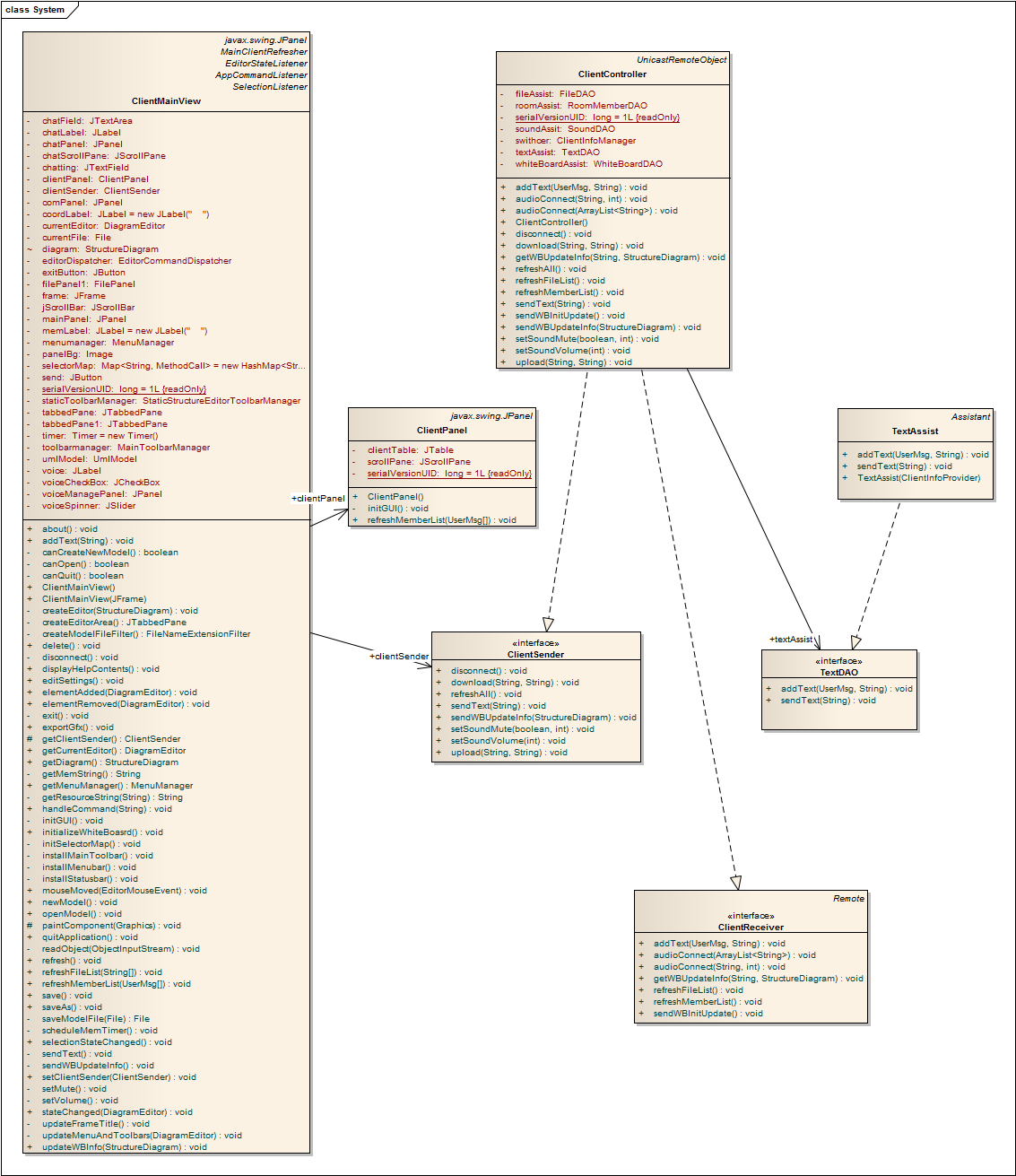
|  |  |  |  |
| --- | --- | --- | --- |
| 文件名称 | 所在目录 | 功能说明 | 重要函数 |
| ClientController | control.client | 负责客户端逻辑控制消息转发 | public void sendWBUpdateInfo(StructureDiagram diagram)  public void getWBUpdateInfo(String senderid,StructureDiagram diagram) |
| WhiteBoardAssist | module.client | 白板核心类 | public void sendWBUpdateInfo(StructureDiagram diagram)  public void getWBUpdateInfo(String senderid,StructureDiagram diagram) |
| ClientMainView | View.client | 白板显示及操作 | public void updateWBInfo(StructureDiagram diagram)  public void sendWBUpdate() |
| EditorStateListener | Whiteboard.view | 白板事件的监听器 | void stateChanged(DiagramEditor editor)  void mouseMoved(EditorMouseEvent event)  void elementAdded(DiagramEditor editor)  void elementRemoved(DiagramEditor editor) |

##### 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名称 | 参数 | 返回值 | 所在文件 | 算法及功能简介 |
| updateWBInfo | StructureDiagram diagram | void | ClientMainView | 根据从服务器所接受到的白板修改信息更新白板内容 |
| sendWBUpdate | 无 | void | ClientMainView | 调用control层方法传输view层对于白板所做的修改 |
| sendWBUpdateInfo | StructureDiagram diagram | void | WhiteBoardAssist | 底层实现传输更新至服务器 |
| getWBUpdateInfo | String senderid,StructureDiagram diagram | void | WhiteBoardAssist | 底层实现接收白板更新内容，并传至view层进行白板的更新 |

#### 日志记录

##### 设计图



##### 功能描述

系统能将会议中参与者的行动记录下来并显示在每个人的文字聊天窗口中，以文字形式展现在所有用户面前。

##### 输入数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 有效性检验规则 |
| 刚加入房间的用户信息 | 一个用户刚刚进入房间时，会自动通知文字聊天窗口 | 此用户会出现在成员列表中 |
| 刚离开房间的用户信息 | 一个用户刚刚离开房间时，会自动通知文字聊天窗口 | 此用户会从成员列表中消失 |
| 用户白板操作内容 | 一个用户进行添加或移除图元时，此操作的信息 | 此操作会体现在白板上 |

##### 输出数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 表现形式 |
| 用户进入房间信息 | 一个用户刚刚进入房间时，会在文字聊天窗口提示 | 文字聊天窗口显示“XX进入了房间” |
| 用户离开房间信息 | 一个用户刚刚离开房间时，会在文字聊天窗口提示 | 文字聊天窗口显示“XX离开了房间” |
| 用户白板操作信息 | 一个用户添加或移除图元时，会在文字聊天窗口提示 | 文字聊天窗口显示“XX添加/移除了图元” |

##### 业务算法和流程

用户加入、退出房间或添加、移除图元时，在服务器转发消息的过程中会自动通知其他客户端的文字聊天模块，使各个用户的文字聊天窗口上显示用户加入、退出房间或添加、移除图元的信息。

##### 数据设计

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据说明 | 数据信息 |
| Sender | 该数据记录发起动作的用户信息。 | 类型为UserMsg的参数，包含name,id等属性。 |

##### 源程序文件说明

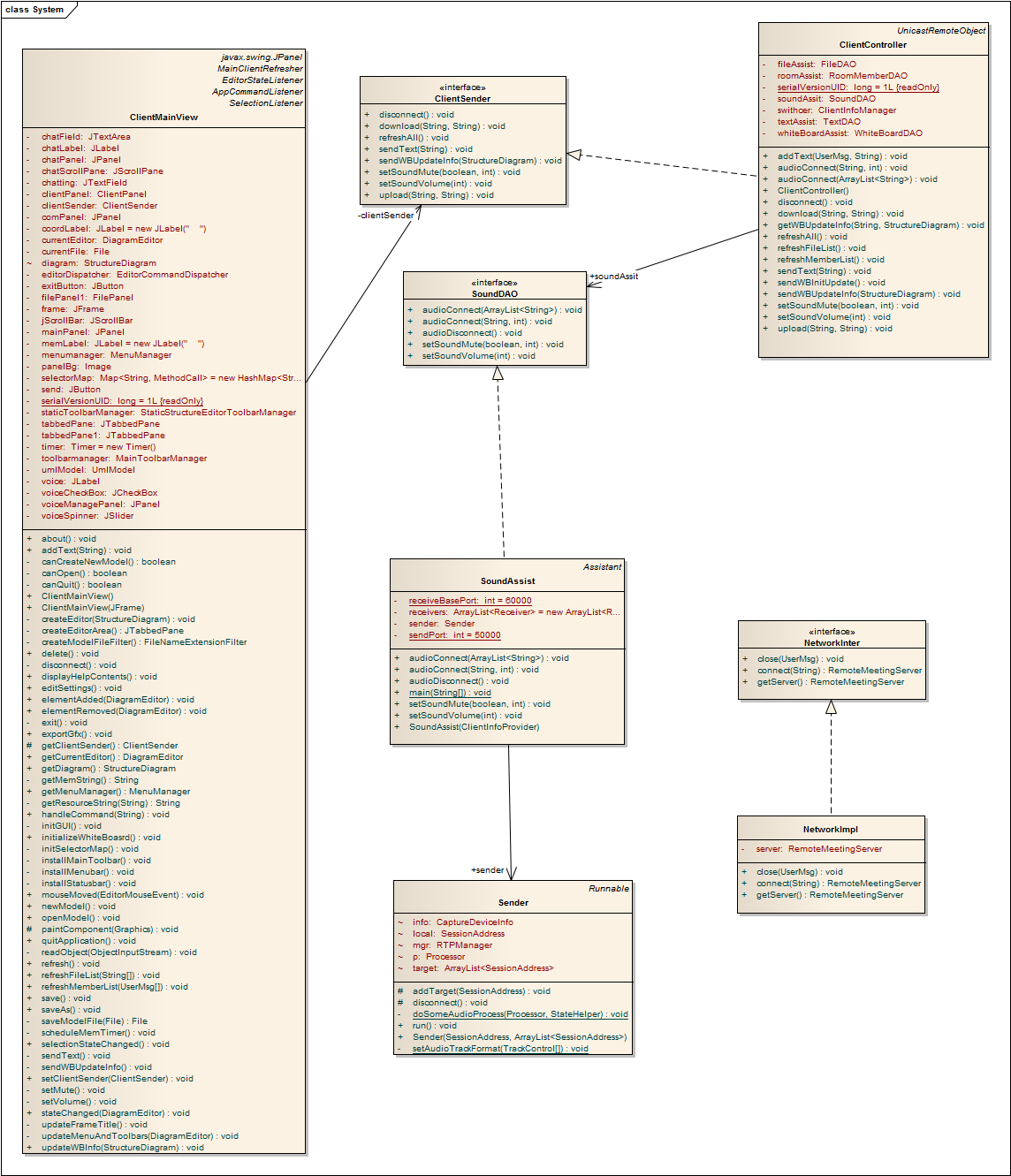
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 文件名称 | | 所在目录 | | 功能说明 | | 重要函数 |
| ClientController | control.client | | 负责客户端逻辑控制消息转发 | | public void addText(UserMsg sender,String message) throws RemoteException; | |
| TextAssist | module.client | | 文字信息核心类 | | public void addText(UserMsg sender, String message) | |
| ClientMainView | View.client | | 负责信息显示的界面 | | public void addText(String text) | |

##### 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名称 | 参数 | 返回值 | 所在文件 | 算法及功能简介 |
| addText | UserMsg sender,String message | void | ClientController | 将信息的转发请求转发给下层 |
| addText | UserMsg sender, String message | void | TextAssist | 根据信息的发送者和信息内容组合成完整信息，转发给View层的addText方法 |
| addText | String text | void | ClientMainView | 在界面中的JTextArea中实际显示所需的文字信息 |

#### 音频聊天

##### 设计图



##### 功能描述

处于同一个会议中的在线用户可以通过语音进行交流：通过麦克风直接说出自己想说的内容，而其他用户能够通过音频接收到其他用户的声音。

##### 输入数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 有效性检验规则 |
| 语音 | 用户在话筒前发出的声音 | 在接收端听到声音 |
| 音量 | 用户希望播放器维持的音量，为0-100间的数字 | 根据输入的音量大小，用户听到的声音大小在静音与最大声音间变动 |

##### 输出数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 表现形式 |
| 语音 | 从发出端发送来的数据 | 声音 |

##### 业务算法和流程

用户加入房间后，建立自己的音频发送端并将房间中所有用户指定为发送目标，并为房间内所有人各准备一个接收端，房间内所有人新加该用户为发送目标，并为其准备一个接收端。一切就绪后开始正常传输声音。

##### 数据设计

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据说明 | 数据信息 |
| 音频 | 使用rtp传输协议封装 | 声音信息 |

##### 源程序文件说明

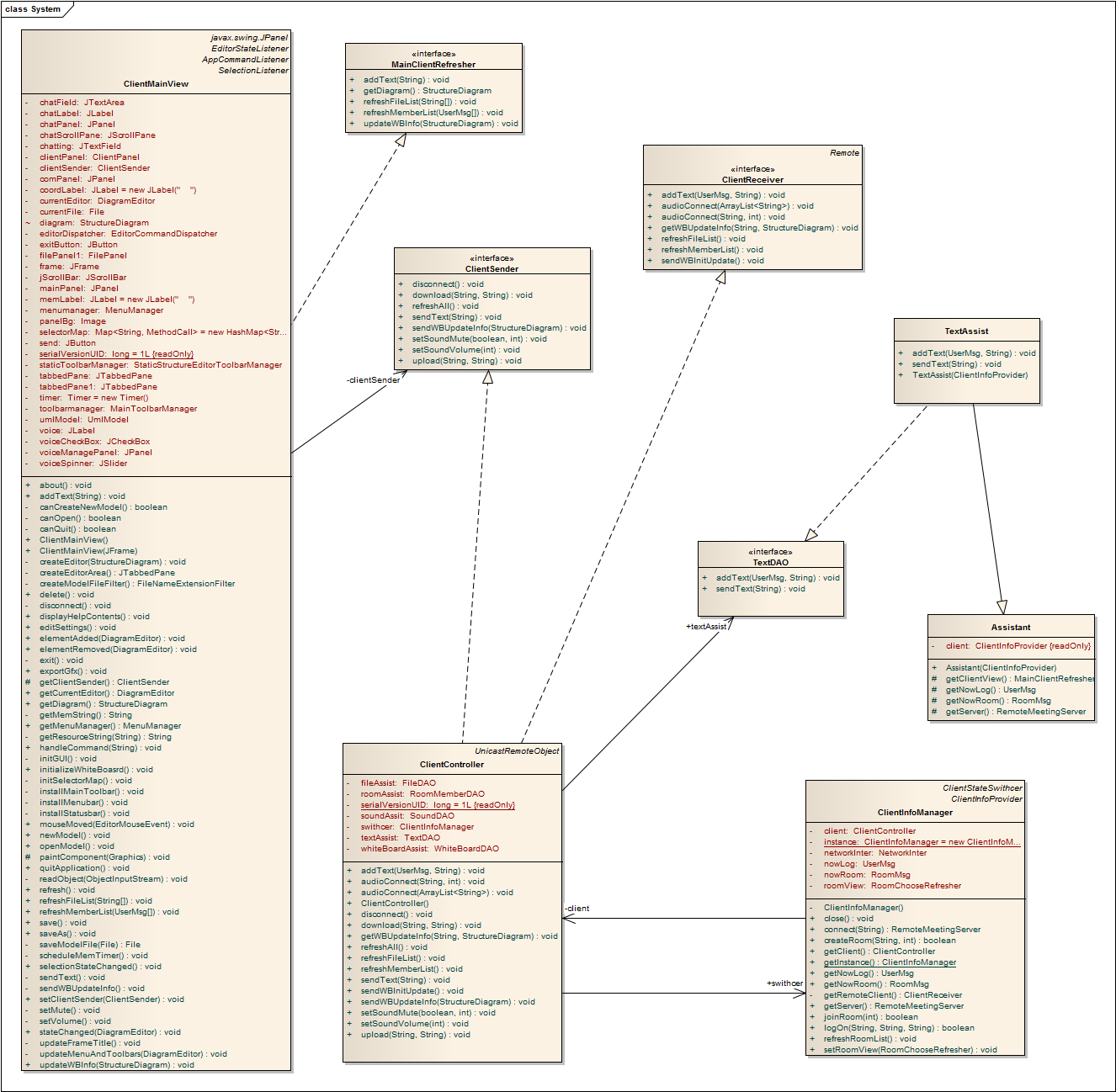
|  |  |  |  |
| --- | --- | --- | --- |
| 文件名称 | 所在目录 | 功能说明 | 重要函数 |
| SoundAssist | Module.Client | 负责管理音频传输 | Audioconnect（） |
| Sender | Module.Client | 负责发送音频数据 | ­—— |
| Receiver | Module.Client | 负责接收音频数据 | —— |

##### 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名称 | 参数 | 返回值 | 所在文件 | 算法及功能简介 |
| Audioconnect（） | String IPlist | Null | Module.Client.soundAssist | 负责在进入房间后进行连接，顺序为：用户加入房间后，建立自己的音频发送端并将房间中所有用户指定为发送目标，并为房间内所有人各准备一个接收端，房间内所有人新加该用户为发送目标，并为其准备一个接收端。一切就绪后开始正常传输声音。 |

#### 文字聊天

##### 设计图



##### 功能描述

用户将自己想要表达的信息通过文字形式发布，显示在文字聊天窗口中，使系统中的其他用户可见。

##### 输入数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 有效性检验规则 |
| 内容 | 用户想要发表的聊天内容 | 不能为空 |
| 确认发送 | 内容输入完毕后确认发送 | 无 |

##### 输出数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 表现形式 |
| 聊天内容 | 用户刚刚发送的文字内容 | 在聊天窗口显示 |

##### 业务算法和流程

##### 数据设计

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据说明 | 数据信息 |
| sender | 发送消息的用户 | UserMsg sender |
| message | 发送消息的内容 | String message |

##### 源程序文件说明

|  |  |  |  |
| --- | --- | --- | --- |
| 文件名称 | 所在目录 | 功能说明 | 重要函数 |
| ClientController | module.client | 负责客户端逻辑控制消息转发 | public void sendText(String text)  public void addText(UserMsg sender, String message) |
| ClientMainView | View.client | 文件列表显示 | public void refresh() |
| TextAssist | Module.clien | 文字传输 | public void sendText(String text)  public void addText(UserMsg sender, String message) |
|  |  |  |  |

##### 函数说明

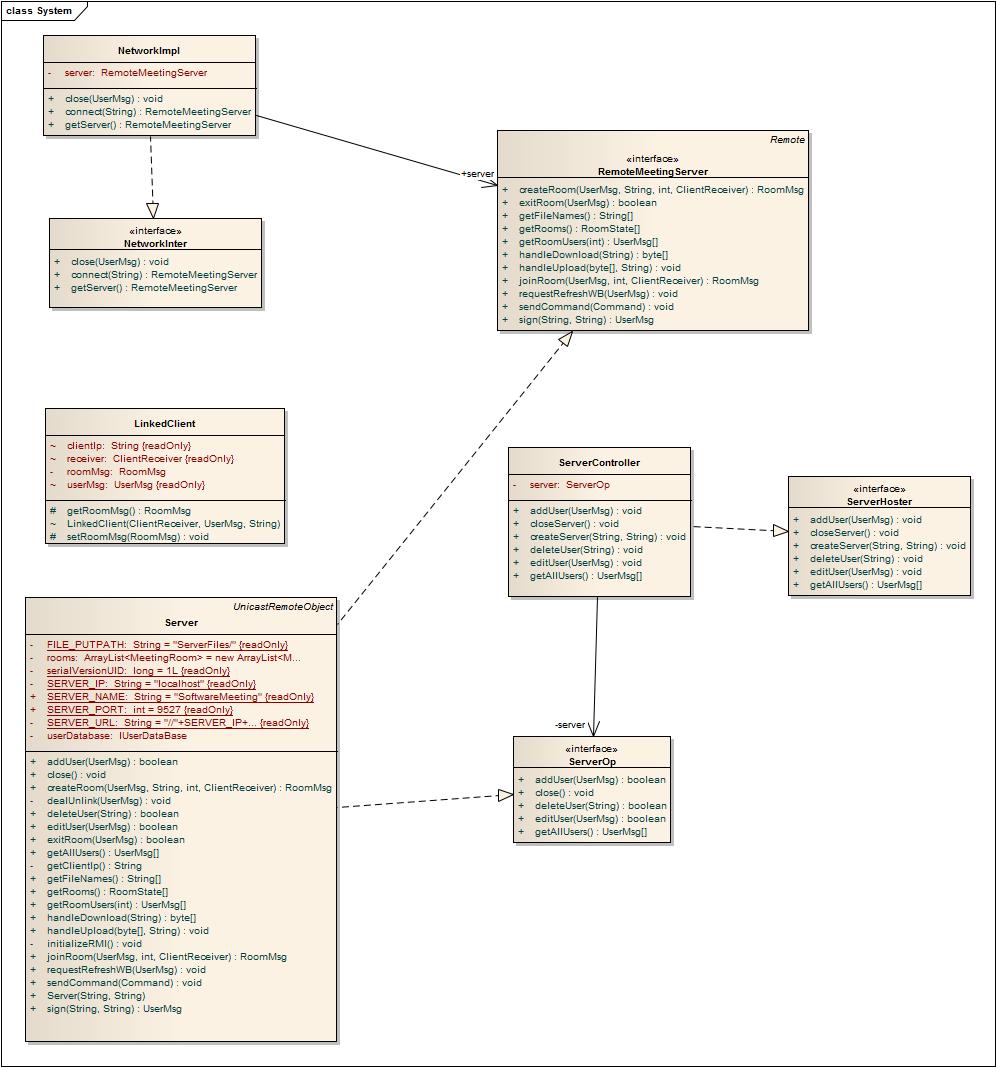
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名称 | 参数 | 返回值 | 所在文件 | 算法及功能简介 |
| addText | UserMsg sender  String message | Void | TextAssist | 调用界面提供的方法将message显示到聊天窗口 |
| sendText | String text | Void | ClientController | 将text发送到服务器 |
| addText | UserMsg sender, String message | void | ClientController | 调用底层的具体实现将message显示到聊天窗口 |

### 服务器

服务器端通过网络接口与客户端连接，负责接收客户端的请求，并向相关的每一个客户端转发请求。服务端采用简化的MVC架构，MV之间不需要连接。下面分别以职能模块和功能模块进行划分并给出详细设计说明。

#### 消息转发

##### 设计图



##### 功能描述

服务器接收来自客户端按照命令模式封装好的命令并对处于同一个房间的所有用户进行转发。

##### 输入数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 有效性检验规则 |
| AddTextCommand | 与文字聊天有关的命令 | 不为空 |
| RefreshListCommand | 与刷新列表有关的命令 | 不为空 |
| WhiteBoardCommand | 与白板图元更新有关的命令 | 不为空 |

##### 输出数据

无

##### 业务算法和流程

服务器与客户端通过网络接口实现连接，网络接口由RMI实现。对于任意一个由客户端发来的命令，服务器可以得到与该客户在同一个房间内的所有客户，并得到他们的远程调用，让其执行该命令。

##### 数据设计

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据说明 | 数据信息 |
| SERVER\_URL | 服务器的URL地址 | private static final String |

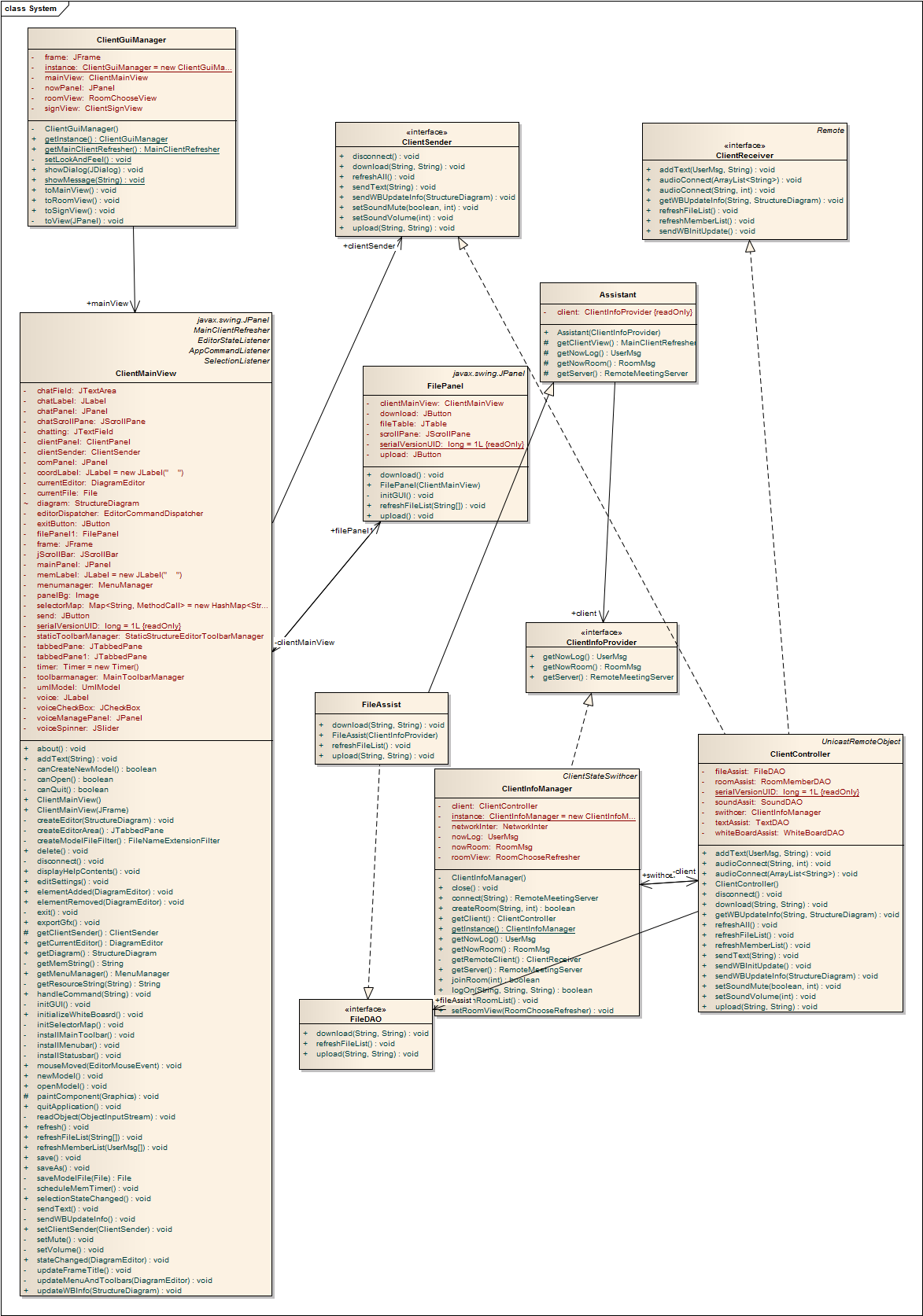
##### 源程序文件说明

|  |  |  |  |
| --- | --- | --- | --- |
| 文件名称 | 所在目录 | 功能说明 | 重要函数 |
| Server | Module.server | 负责转发来自客户端的各种命令 | private void initializeRMI()  public RoomMsg joinRoom(UserMsg user, int roomId, ClientReceiver clientReceiver)  public RoomState[] getRooms()  public void sendCommand(Command c) |
| MeetingRoom | Module.server | 处理各种与房间有关的事情，包括对房间内现有用户的相关操作 | protected UserMsg[] getRoomUsers()  protected synchronized void addCommand(Command c)  protected int getRoomId()  protected RoomMsg getRoomMsg() |

##### 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名称 | 参数 | 返回值 | 所在文件 | 算法及功能简介 |
| sendCommand | Command c | void | Server | 添加该命令到某一个房间，不管是什么命令 |
| initializeRMI | None | Void | Server | 初始化RMI，将服务器的URL地址绑定到server类 |
| addCommand | Command c | Void | MeetingRoom | 添加一个命令到某个房间，对该房间内的每一个用户的远程引用执行该命令 |

#### 资料共享



##### 功能描述

用户可以将本地的文件共享在系统中，供所有用户下载；也可以自由下载其他用户已经共享的文件资料。

##### 输入数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 有效性检验规则 |
| 文件名 | 用户要下载的文件名 | 不为空且存在 |
| 文件内容 | 用户上传的文件内容 | 无 |

##### 输出数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 表现形式 |
| 文件列表 | 上传文件后服务器所有的新文件列表 | 列表刷新 |
| 文件 | 从服务器下载的文件 | 在之前选择的文件夹出现选择下载的文件 |

##### 业务算法和流程

上传文件：服务器接收客户端传来的二进制流并将文件保存在固定目录

下载文件：服务器根据客户端选择下载的文件名将文件解析成二进制流，加以文件的名称等信息发送给客户端。

##### 数据设计

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据说明 | 数据信息 |
| fileName | 上传/下载文件的名称 | String fileName |
| content | 被解析成二进制流的文件 | byte content[] |

##### 源程序文件说明

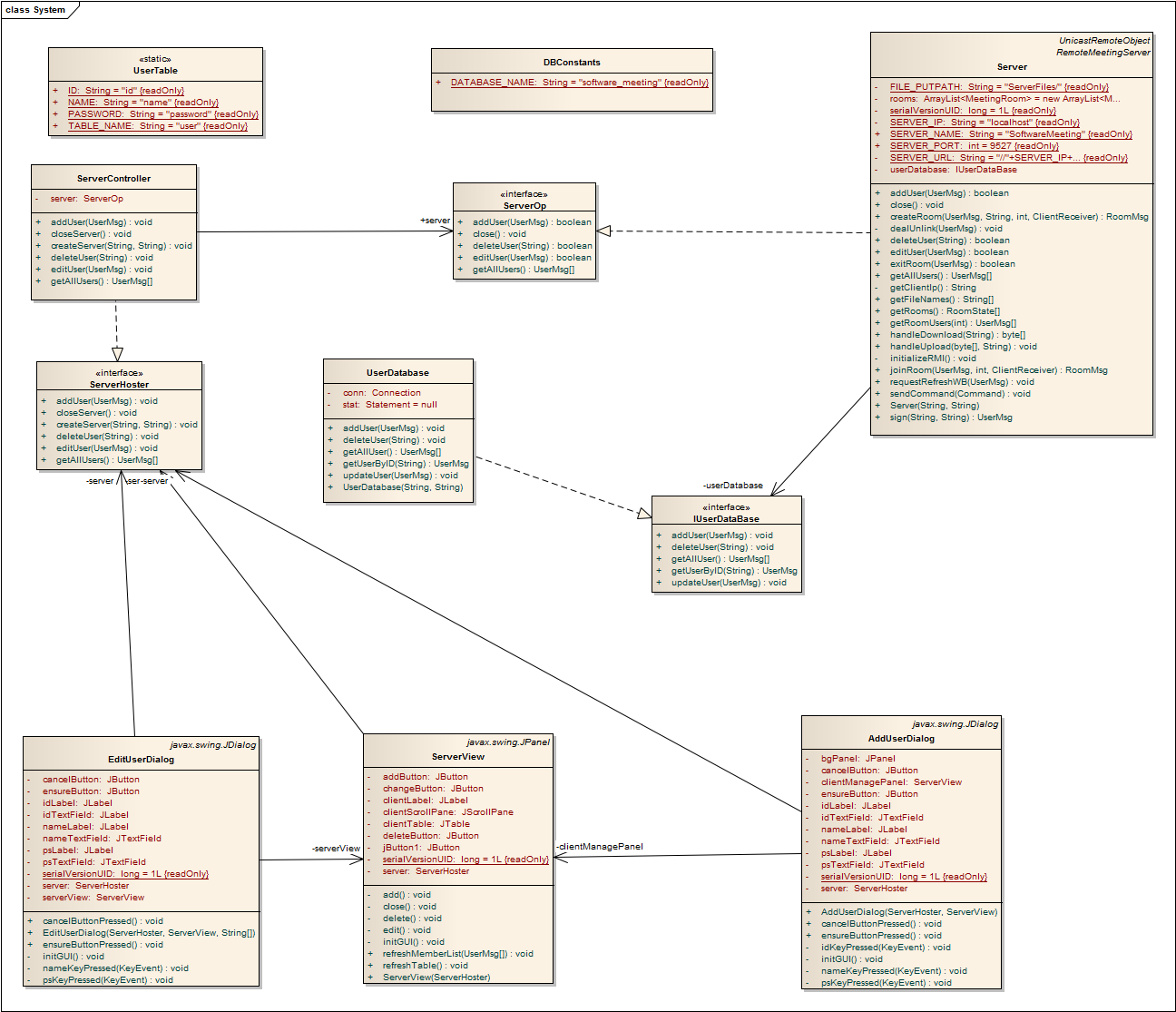
|  |  |  |  |
| --- | --- | --- | --- |
| 文件名称 | 所在目录 | 功能说明 | 重要函数 |
| Server | Module.server | 负责服务器端核心的逻辑操作 | public void handleUpload(byte[] content, String fileName)  public byte[] handleDownload(String fileName)  public String[] getFileNames() |

##### 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名称 | 参数 | 返回值 | 所在文件 | 算法及功能简介 |
| handleUpload | byte[] content  String fileName | Void | Server | 处理客户端的上传文件请求，接收传来的二进制流并保存到服务器目录下 |
| handleDownload | String fileName | byte[] | Server | 根据用户要下载的文件名将目录下的文件解析成二进制流并传回客户端 |

#### 账户管理

##### 设计图



##### 功能描述

系统管理员可以对于该软件用户的账户信息在后台进行增删查改，并且可以查看每个技术人员的资料。

##### 输入数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 有效性检验规则 |
| 要增删查改的用户 | 要增删查改的用户，在用户列表中直接选择即可 | 不为空 |
|  |  |  |

##### 输出数据

|  |  |  |
| --- | --- | --- |
| 数据名称 | 简要介绍 | 表现形式 |
| 用户信息 | 选中的用户信息 | 弹出窗口以文字显示 |

##### 业务算法和流程

##### 数据设计

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据说明 | 数据信息 |
| conn | 对数据库连接的引用 | Connection conn |
| stat | 执行数据库查询操作的对象 | Statement stat |

##### 源程序文件说明

|  |  |  |  |
| --- | --- | --- | --- |
| 文件名称 | 所在目录 | 功能说明 | 重要函数 |
| UserMsg | Util.message | 用户信息 | public boolean equals(Object obj) |
| UserDatabase | Module.server | 对数据库的具体操作 | public void addUser(UserMsg user)  public void deleteUser(String id)  public void updateUser(UserMsg user)  public UserMsg getUserByID(String id)  public UserMsg[] getAllUser() |
| DBConnect | Module.server | 连接本地mysql数据库 | public static void init(String sname,String password) |
| DBConstants | Module.server | 定义了一些与数据库表相关的常量 | 无 |

##### 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名称 | 参数 | 返回值 | 所在文件 | 算法及功能简介 |
| addUser | UserMsg user | Void | UserDatabase | 添加一个用户 |
| deleteUser | String id | Void | UserDatabase | 根据用户id删除一个用户 |
| updateUser | UserMsg user | Void | UserDatabase | 更新用户资料 |
| getUserByID | String id | UserMsg | UserDatabase | 通过用户id得到一个用户资料 |
| getAllUser | None | UserMsg[] | UserDatabase | 得到所有用户列表 |
| getConnection | none | Connection | DBConnect | 得到与数据库连接的一个实例 |
| init | String sname  String password | Void | DBConnect | 通过jdbc连接mysql数据库 |

# 接口设计

## 内部接口

### 接口说明

#### Controller层

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 接口内容 |
| UserIdentifier | 供登录界面使用，提供验证用户名密码的方法。 | /\*\*验证输入的用户名和密码是否正确。  \* @param sname  \* @param password  \* @return 是否登录成功  \* @throws ServerNotActiveException  \* @throws RemoteException \*/  public boolean sign(String sname,String password, String ip) throws RemoteException, ServerNotActiveException; |
| RoomChooser | 用户在房间选择界面里的操作接口。提供用户加入房间的方法。 | /\*\*登录的用户创建一个新的房间  \* @param name 房间名称  \* @param maxMem 最大人数  \* @return 创建是否成功 \*/  public boolean createRoom(String name,int maxMem);  /\*\*刷新房间列表。  \* @throws RemoteException \*/  public void refreshRoomList() throws RemoteException; |
| ClientSender | 客户端用于发送信息的接口。此接口负责与服务器接口进行连接。 | /\*\*上传一个文件到服务器。  \* @param filePath 上传的文件在上传者机器的地址  \* @param filename 文件名  \* @throws RemoteException \*/  public void upload(String filePath,String filename) throws RemoteException;    /\*\*下载一个文件到本机。  \* @param putPath 要下载的文件放置在本机的地址  \* @param filename 文件名  \* @throws RemoteException \*/  public void download(String putPath,String filename) throws RemoteException;    /\*\*发送一条文字信息。  \* @param text 消息的内容。  \* @throws RemoteException \*/  public void sendText(String text) throws RemoteException;    public void refreshAll() throws RemoteException;    public void disconnect() throws RemoteException;    /\*\*调节话筒音量  \* @param v \*/  public void setSpeakerVolume(int v);    /\*\*调节话筒静音  \* @param v \*/  public void setSpeakerMute(boolean mute);    /\*\*调节音响音量  \* @param v \*/  public void setSoundVolume(int v);    /\*\*调节音响静音  \* @param v \*/  public void setSoundMute(boolean mute);    /\*\*  \* 发送白板的更新信息  \*/  public void sendWBUpdateInfo(StructureDiagram diagram) throws RemoteException; |
| ClientReceiver | 服务器端回调客户端所用的接口。客户端必须要实现这个接口。 | /\*\*刷新文件列表。 \*/  public void refreshFileList() throws RemoteException;    /\*\*刷新房间成员列表。 \*/  public void refreshMemberList() throws RemoteException;    /\*\*在文字信息窗口中添加一条信息。 \*/  public void addText(UserMsg sender,String message) throws RemoteException;    /\*\*一个人刚进来，和其他所有人连接。  \* @param ip 其他人的ip\*/  public void audioConnect(ArrayList<String> ip)throws RemoteException;    /\*\*大家和一个刚进来的人连接。  \* @param ip \*/  public void audioConnect(String ip,int No)throws RemoteException;    /\*\*  \* 更新白板内容  \* @throws RemoteException  \*/  public void getWBUpdateInfo(StructureDiagram diagram) throws RemoteException; |
| ServerHoster | Controller层向view层提供的接口 | /\*\*创建可用的服务器  \* @throws RemoteException  \* @throws MalformedURLException  \* @throws AlreadyBoundException  \*/  public void createServer(String sname, String password) throws RemoteException,  MalformedURLException, AlreadyBoundException;      /\*\*关闭服务器 \*/  public void closeServer();    /\*\*按照给出的信息添加一个用户。  \* @param usr \*/  public void addUser(UserMsg usr);    /\*\*按照给出的信息，把对应id的用户信息以usr信息覆盖。  \* @param usr \*/  public void editUser(UserMsg usr);    /\*\*删掉对应id的用户。  \* @param id \*/  public void deleteUser(String id);    public UserMsg[] getAllUsers() throws SQLException; |

#### Module层

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 接口信息 |
| ClientInfoProvider | /\*\*让Assist类查看客户端信息用的接口。  \* @author Playeye \*/ | /\*\* @return 远程的服务器引用 \*/  public RemoteMeetingServer getServer();  /\*\* @return 现在登录着的用户的信息 \*/  public UserMsg getNowLog();    /\*\* @return 现在所在的房间的信息 \*/  public RoomMsg getNowRoom(); |
| ClientStateSwithcer |  | /\*\*将本客户端连接到服务器。  \* @param clientReceiver 客户端远程引用  \* @param ipString 服务器所在IP  \* @return 连接到的服务器对象  \* @throws MalformedURLException 无效的IP  \* @throws RemoteException 远程连接出错  \* @throws NotBoundException 找不到服务器  \* @throws ServerNotActiveException  \*/  public RemoteMeetingServer connect(String ipString) throws MalformedURLException, RemoteException, NotBoundException, ServerNotActiveException;    /\*\*关闭此客户端和服务器的连接，并在关闭前告知服务器自己的编号。  \* @throws RemoteException \*/  public void close() throws RemoteException;    /\*\*让用户登录  \* @param sname  \* @param password  \* @return 登录的用户  \*/  public boolean logOn(String sname, String password, String ip)throws RemoteException,ServerNotActiveException;    /\*\*对应用户加入对应id的房间。  \* @param roomId  \* @return 是否加入成功  \* @throws RemoteException \*/  public boolean joinRoom(int roomId) throws RemoteException;    /\*\*对应用户创建房间。  \* @param id 不可重复的id  \* @param name 房间名  \* @param maxMem 最大人数  \* @return 创建的房间信息  \* @throws RemoteException \*/  public boolean createRoom(String name,int maxMem) throws RemoteException;    public void refreshRoomList() throws RemoteException; |
| RemoteMeetingServer | 服务器端供客户端远程调用的接口。 | /\*\*用户登录系统，获得自身的信息。  \* @param sname 用户名  \* @param password 密码  \* @return 用户的信息  \* @param clientReceiver 客户端的远程引用  \* @throws RemoteException \*/  public UserMsg sign(String sname,String password) throws RemoteException, ServerNotActiveException;    /\*\*用户加入一个房间  \* @param user 用户信息  \* @param roomId 房间号  \* @return 加入的房间信息 \*/  public RoomMsg joinRoom(UserMsg user, int roomId ,ClientReceiver clientReceiver)throws RemoteException;  /\*\*用户新建一个房间  \* @param user 用户信息  \* @param name 房间名  \* @param maxMem 最大人数  \* @return 创建的房间信息  \*/  public RoomMsg createRoom(UserMsg user, String name, int maxMem,ClientReceiver clientReceiver)throws RemoteException;    /\*\*让某用户退出房间，返回大厅  \* @param user 用户信息  \* @param room 房间信息\*/  public boolean exitRoom(UserMsg user)throws RemoteException;    /\*\*将一个命令对象传输到服务器去。  \* @param c \*/  public void sendCommand(Command c) throws RemoteException;  /\*\*  \* 处理客户端的文件下载操作  \* @param fileName 文件名称  \* @return 文件内容，byte[]  \* @throws RemoteException  \*/  public byte[] handleDownload(String fileName) throws RemoteException;  /\*\*  \* 处理客户端的文件上传操作  \* @param content 文件内容  \* @param fileName 文件名称  \*/  public void handleUpload(byte[] content, String fileName)throws RemoteException;  /\*\*  \* 取得服务器所有的文件名称列表  \* @return 文件名称列表  \*/  public String[] getFileNames()throws RemoteException;    public RoomState[] getRooms()throws RemoteException;    public UserMsg[] getRoomUsers(int roomId)throws RemoteException; |
| FileDAO | 控制层访问文件共享子系统所用接口。 | /\*\*上传一个文件到服务器。  \* @param filePath 上传的文件在上传者机器的地址  \* @param putPath 在服务器上放置文件的地址  \* @throws RemoteException \*/  public void upload(String filePath,String putPath) throws RemoteException;    /\*\*下载一个文件到本机。  \* @param fromPath 要下载的在服务器上的放置地址  \* @param putPath 下载到下载者机器的路径  \* @throws RemoteException \*/  public void download(String fromPath,String putPath) throws RemoteException;      /\*\*将底层的文件列表刷新后更新到界面里。  \* @throws RemoteException \*/  public void refreshFileList() throws RemoteException; |
| RoomMemberDAO | 控制层访问文件共享子系统所用接口。 | / /\*\*刷新房间成员列表。  \* @throws RemoteException \*/  public void refreshMemberList() throws RemoteException;    public void exitRoom()throws RemoteException; |
| TextDAO | 控制层访问文字聊天子系统所用接口。 | /\*\*发送一条文字信息。  \* @param sender 发送者  \* @param text 消息的内容。  \* @throws RemoteException \*/  public void sendText(String text) throws RemoteException;      /\*\*添加一条文字消息到窗口。  \* @param sender 发送者  \* @param message 信息内容  \*/  public void addText(UserMsg sender, String message) ; |
| IUserDataBase | 数据库向控制层提供的接口,对用户数据库的一系列操作 | /\*\*  \* 增加一个user  \* @param user  \* @throws SQLException  \*/  public void addUser(UserMsg user) throws SQLException;  /\*\*  \* 根据删除一个User  \* @param id  \* @throws SQLException  \*/  public void deleteUser(String id) throws SQLException;  /\*\*  \* 修改User资料，其中id是不能被修改的  \* @param user  \* @throws SQLException  \*/  public void updateUser(UserMsg user) throws SQLException;  /\*\*  \* 根据User的id取的user的相关信息  \* @param id  \* @throws SQLException  \*/  public UserMsg getUserByID(String id) throws SQLException;  /\*\*  \* 返回数据库中所有的用户信息  \* @return  \* @throws SQLException  \*/  public UserMsg[] getAllUser()throws SQLException; |
| WhiteBoardDAO | 控制层访问电子白板子系统所用接口。 | /\*\*  \* 发送更新白板上的所有内容  \* @throws RemoteException  \*/  public void sendWBUpdateInfo(StructureDiagram diagram) throws RemoteException;    /\*\*  \* 获得白板更新  \*/  public void getWBUpdateInfo(StructureDiagram diagram); |
| SoundDAO | 对音频操作的接口 | public void audioConnect(ArrayList<String> ip);    public void audioConnect(String ip,int No);  public void audioDisconnect();    /\*\*调节话筒音量  \* @param v \*/  public void setSpeakerVolume(int v);    /\*\*调节话筒静音  \* @param v \*/  public void setSpeakerMute(boolean mute);    /\*\*调节音响音量  \* @param v \*/  public void setSoundVolume(int v);    /\*\*调节音响静音  \* @param v \*/  public void setSoundMute(boolean mute); |
| ServerOp | 控制层向view层提供的接口 | public void close();  /\*\*按照给出的信息添加一个用户。  \* @param usr \*/  public boolean addUser(UserMsg usr);    /\*\*按照给出的信息，把对应id的用户信息以usr信息覆盖。  \* @param usr \*/  public boolean editUser(UserMsg usr);    /\*\*删掉对应id的用户。  \* @param id \*/  public boolean deleteUser(String id);  public UserMsg[] getAllUsers() throws SQLException ; |

#### View层

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 接口信息 |
| RoomChooseRefresher | 为module层提供的更改view层的接口，用于刷新当前房间列表 | /\*\*刷新选择房间界面\*/  public void refreshRoomlist(RoomState[] rooms) |
| MainClientRefresher | 为module层提供的更改view层的接口，用于刷新客户端的文件列表、在线用户列表、白板 | /\*\*在文字聊天窗口里添加一句话  \* @param text \*/  public void addText(String text)  /\*\*注意！参数现在还不知道，所以你以后自己改 \*/  public void refreshFileList(String[] fileNames);    /\*\*更新房间成员列表的显示。  \* @param members \*/  public void refreshMemberList(UserMsg[] members)    /\*\*  \* 更新白板内容  \* @param diagram  \*/  public void updateWBInfo(StructureDiagram diagram) |

#### 网络传输部分

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 接口信息 |
| NetworkInter | 网络的接口实现 | /\*\*将本客户端连接到服务器。  \* @param clientReceiver 客户端远程引用  \* @param ipString 服务器所在IP  \* @return 连接到的服务器对象  \* @throws MalformedURLException 无效的IP  \* @throws RemoteException 远程连接出错  \* @throws NotBoundException 找不到服务器  \* @throws ServerNotActiveException  \*/public RemoteMeetingServer  connect(String ipString)  /\*\*关闭此客户端和服务器的连接，并在关闭前告知服务器自己的编号。  \* @throws RemoteException \*/public void close(UserMsg nowLog)  /\*\*取得对远程服务器的引用\*/  public RemoteMeetingServer getServer() |

#### 命令部分

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 接口信息 |
| Command | 在服务器与客户端之间传输，统一定义的命令行为接口 | /\*\*在服务端调用该方法，让其中一个客户端来执行此命令。  \* @param client  \* @throws RemoteException \*/  public void excuteOnClient(ClientReceiver client) throws RemoteException;    /\*\* @return id of the effecting room \*/  public int getRoomId(); |

# 数据库设计

由于只有账户管理用例涉及到数据库，所以数据库设计较简单。

|  |  |  |
| --- | --- | --- |
| 表名称 | 属性名 | 类型 |
| User | ID | Varchar(20) |
| name | varchar(20) |
| password | varchar(20) |

# 系统安全保密设计

## 说明

由于本软件针对技术人员讨论软件技术问题而开发，为防止与所讨论软件有关的内容外流，需要考虑安全保密问题。

## 设计

### 身份验证部分

* 每一个用户账号由管理员发放，用户需要账号、密码登录才能进入会议房间
* 不在同一个房间进行会议的用户无法得知其他会议的进行状况

### 音频聊天部分

* 每个用户在进入房间后获得房间中人IP列表，与房间内所有人开始连接。
* 用户可以调节声音播放的音量大小
* 音频聊天功能借用jmf实现

# 系统性能设计

1. 允许多个房间同时进行多场会议
2. 每个房间在线人数不超过8人
3. 若中途网络断开则退出会议，并需要重新登录和重新进入会议
4. 声音经传输在另一用户听到后尽量不失真，减小杂音
5. 对于共享白板，修改的内容必须在2秒内同步到其他用户的屏幕上
6. 对于文件共享，若上传/下载到一半时出现网络异常，至少能恢复到上传前的状态

# 系统出错处理

* 用户掉线或服务器出错时弹出窗口提示如下：

