|  |
| --- |
| 主要算法与设计： |
| 1. **gui\_menu 图形化界面**    1. **界面设计**   采用Swing-UI-designer设计主要界面，并增加少许Swing语句(如setTitle, setAlwaysOnTop)   * 1. **ServerMainMenu 服务端主界面**   **a. 成员**  类的成员除Swing的窗口组件(如JPanel, JButton)外，我设计了一些与功能核心相关的成员，其作用如表2.1所示:   |  |  | | --- | --- | | 成员类型-名称 | 作用 | | int port | 记录服务端端口号 | | SocketServer server | 一个包装了(serverSocket类成员[在socket包中])的对象 |   **表2.1 ServerMainMenu核心成员及作用**  **b. 构造函数**  构造函数负责将server置null，并指定port为DEFAULT\_PORT(全局常量)，并设置一些窗口相关属性与监听器。  **c. 监听器**  每个按钮都对应一个监听器(均采用鼠标单击监听)，用来执行点击后的操作。具体按钮、触发条件及单击后的作用和设计如表2.2所示：   |  |  |  | | --- | --- | --- | | 按钮 | 条件 | 作用和设计 | | 运行服务端  runServerButton | server为null时 | 通过port来new一个SocketServer对象 并调用SocketServer的start开启server | | 服务端状态  showStatusButton | server开启时(非null) | 调用SocketServer的showLoaclStatus展现服务端连接状态 | | 关闭服务端  closeServerButton | server开启时 | 调用SocketServer的exit关闭server对象，并置server为null | | 更新端口号  confirmPortButton | server为null时 | 读取端口文本框(portInput)，来更新port成员 |   **表2.2 ServerMainMenu窗口按钮及按下后的作用**   * 1. **ClientMainMenu 客户端主界面**   **a. 成员**  类的成员除Swing的窗口组件(如JPanel, JButton)外，我设计了一些与功能核心相关的成员，其作用如表2.3所示:   |  |  | | --- | --- | | 成员类型-名称 | 作用 | | String serverName; | 记录要连接的服务端的地址 | | int port; | 记录要连接的服务端的端口号 | | SocketClient client; | 一个包装了(serverSocket类成员[在socket包中])的对象 |   **表2.3 ClientMainMenu核心成员及作用**  **b. 构造函数**  构造函数负责将client置null，并指定serverName为”localhost”，port为DEFAULT\_PORT(全局常量)，并设置一些窗口相关属性与监听器。  **c. 监听器**  每个按钮都对应一个监听器(均采用鼠标单击监听)，用来执行点击后的操作。具体按钮、触发条件及单击后的作用和设计如表2.4所示：   |  |  |  | | --- | --- | --- | | 按钮 | 条件 | 作用和设计 | | 开启客户端  runServerButton | client为null时 | 通过port和serverName来new一个SocketClient对象 | | 身份验证  loginCheckButton | client开启时 | 打开登录弹窗 | | 关闭客户端  closeServerButton | client开启时 | 调用SocketClient的exit关闭client对象，并置client为null | | 更新地址与端口号  confirmPortButton | client为null时 | 读取端口文本框(portInput)与地址文本框(serverNameInput)，来更新port与serverName成员 | | 发送信息/本地文件  sendMessageButton | client开启时且client已登陆 | 调用SocketClient的sendMessage发送信息给服务端(通过SocketClient的isLogin()查询登录状态) |   **表2.4 ClientMainMenu窗口按钮及按下后的作用**   * 1. **Login 登录弹窗**   **a. 成员及构造函数**  类的成员只有Swing的窗口组件(如JPanel, JButton)，作为过渡用的弹窗，没有其他核心成员。  **b. 构造函数**  构造函数负责负责传入一个SocketClient对象，用于后续指定客户端进行登录操作，并设置一些窗口相关属性与监听器。  **c. 监听器**  每个按钮都对应一个监听器(均采用鼠标单击监听)，用来执行点击后的操作。具体按钮、触发条件及单击后的作用和设计如表2.5所示：   |  |  |  | | --- | --- | --- | | 按钮 | 条件 | 作用和设计 | | 登录  loginButton | 无 | 先将用户名框与密码框的信息结合为User对象，再调用User类的toString转化为字符串，最后通过SocketClient的loginOneTime传输给服务端。如果接到登录成功的反馈(表现为loginOneTime的返回值)，则调用SocketClient的setHasLogin将client的登录状态置为true | | 取消登录  cancelButton | 无 | 调用SocketClient的setHasLogin将client的登录状态置为false，并关闭窗口 |   **表2.5 Login窗口按钮及按下后的作用**   * 1. **Util工具类设计**   工具类存放了一些上述窗口类共同用到的东西，具体有表2.6中的三个函数：   |  |  |  | | --- | --- | --- | | 函数名 | 作用范围 | 作用和设计 | | freshLog  刷新日志 | JTextArea文本框 | 当文本框内的行数大于等于MAX\_ROW(全局变量)时，取文本框第二行开始的子串替换文本框原来的内容(删去第一行) | | logAppend  添加日志 | 大部分menu内的监听器，以及socket包下的核心类 | 接收字符串与JTextArea对象，当JTextArea对象存在时，为文本框增加内容，并在最后调用freshLog防止日志过长。这个函数设置来提醒使用者已执行的操作。 | | transmitFile  传输文件 | 在SocketClient类中的sendMessage中使用 | 接收参数以指定文件位置，并打开文件输入流。创建StringBuilder并在开头附上文件标识前缀"\_\_\_\_FILE\_\_\_\_"，并依次以字节形式将文件内容依次写入字符串。最后返回由StringBuilder转换的String对象。 |   **表2.6 Util工具类中函数的作用及作用范围**   1. **socket 核心功能类(并非java.net.socket)**    1. **SocketServer 服务端核心类**   **a. 成员**  类的成员主要由表3.1组成：   |  |  | | --- | --- | | 成员类型-名称 | 作用 | | ServerSocket serverSocket | Java.net.socket的原装对象，服务端的核心成员 | | UserData database | 自定义的数据库对象，位置在data包内，通过HashSet  存储由User对象转化的String对象 | | DataInputStream in | 输入流对象，由于设置为1对1的C/S服务器，因此对接用的Socket输入流设置为类的成员 | | DataOutputStream out | 输出流对象，由于设置为1对1的C/S服务器，因此对接用的Socket输出流设置为类的成员 | | boolean isConnected | 用于判断服务器是否处于连接状态，仅用于获取服务器状态时输出 | | JTextArea consoleLog | 文本框对象，由ServerMainMenu提供 |   **表3.1 SocketServer的成员**  **b. 继承关系及函数重载**  继承了线程类(Thread)，一开始用于控制台界面同时开启客户端与服务端，并为一对多C/S服务器做基础(虽然并未实现)。  重写了Thread类中的运行方法(run)，改为直接调用服务器运行函数(runServer)。  **c. 构造函数**  构造函数用于通过默认端口号初始化以上成员(in与out除外，因为此时并没有与客户端相连)。  **d. 主流程启动函数**  为了简化C/S对接的问题，在服务端上我采用了固定流程运作的设计。当服务端被建立并运行时，将不停进入循环，再循环内，首先尝试接收接收服务端的连接，如果超过了最大等待时间(全局变量WAITING\_TIME)，将退出循环，执行关闭操作。如果成功接收到连接则进入下一步：检查登录(一直尝试接收由用户名与密码组成的字符串，并验证database是否包含此信息串，返回boolean类型数据给客户端表示验证结果)。如验证成功则进入下一步：不停接收信息。  虽然主要的流程是固定的，但提供了关闭服务端的按钮，可以做到一键退出上述流程，并通过try-catch语句块保证了中断后程序仍能正常运作。    **e. 主要过程函数**  三个在d中提到的不断循环的过程函数的作用如表3.2所示：   |  |  |  | | --- | --- | --- | | 函数 | 条件 | 作用 | | acceptClient  不停尝试接收客户端连接 | 无 | 调用jdk原装的accept尝试与客户端建立连接。如果成功建立连接，则以该客户端为in和out赋值。如果在过程中因长时间无连接或连接对象已断开，则将返回null值(后续过程中一旦传入null值，函数就自动结束，进入下一个主流程循环) | | checkLogin  不停检查登录 | 传入非null的server | 一直尝试通过checkLoginOneTime验证database是否包含此信息串，返回boolean类型数据给客户端表示验证结果。 | | checkLoginOneTime  检查一次登录 | 无 | 接收由用户名与密码组成的字符串，调用Userdata类的函数验证database里是否有此串 | | receiveMessage  不停接受信息 | 传入非null的server | 不停接收信息，并附加一些额外日期信息并显示在日志文本框中 |   **表3.2 SocketServer的主要过程函数**  **f. 其他函数**  除去前述函数外，还有三个配合按钮的函数，如表3.3所示：   |  |  | | --- | --- | | 函数名 | 作用 | | showLocalStatus  显示服务端本体状态 | 展示服务端端口号与当前连接状态 | | showConnectStatus  显示服务端连接状态 | 展示服务端主机地址与客户端主机地址 | | exit  退出 | 执行in，out，serverSocket的close()以关闭全部连接 |   **表3.3 SocketServer的其他函数**     * 1. **SockeClient 客户端核心类**   **a. 成员**  类的成员主要由表3.4组成：   |  |  | | --- | --- | | 成员类型-名称 | 作用 | | Socket client | Java.net.socket的原装对象，客户端的核心成员 | | DataInputStream in | 输入流对象，由于设置为1对1的C/S服务器，因此对接用的Socket输入流设置为类的成员 | | DataOutputStream out | 输出流对象，由于设置为1对1的C/S服务器，因此对接用的Socket输出流设置为类的成员 | | boolean hasLogin | 用于判断客户端是否已登录，即是否通过服务端验证密码 | | JTextArea consoleLog | 文本框对象，由ClientMainMenu提供 |   **表3.4 SocketClient的成员**  **b. 继承关系及函数重载**  继承了线程类(Thread)，一开始用于控制台界面同时开启客户端与服务端，并为一对多C/S服务器做基础(虽然并未实现)。  重写了Thread类中的运行方法(run)，改为直接调用服务器运行函数(runClient)。  **c. 构造函数**  构造函数用于通过默认地址与默认端口号初始化以上成员(如果成功连接到服务端，则也对in与out进行初始化，将输入输出流指向服务端。  **d. 主要函数**  出于图形化界面的设计，在客户端上我采用了相对灵活设计。当客户端被建立并运行时，不再进入循环，而是完全通过按钮来调动功能，其中主要涉及的函数如表3.5所示：   |  |  | | --- | --- | | 函数名 | 作用与设计 | | loginOneTime  进行一次登录 | 将登录的信息通过out.writeUTF传递给服务端，并返回来自服务端的验证(return in.readBoolean()) | | sendMessage  向服务端发送一条信息 | 首先判断信息是否为地址(长度>2且第二位为’:’第三位为”\”)。如果是，调用transmitFile给服务端发送文件，否则发送信息 | | exit  退出 | 执行in，out，client的close()以关闭全部连接 |   **表3.5 SocketClient的主要函数**  **e. 其他函数**  除去前述函数外，还有三个配合按钮的函数，如表3.3所示：   |  |  | | --- | --- | | 函数名 | 作用 | | showLocalStatus  显示服务端本体状态 | 展示服务端端口号与当前连接状态 | | showConnectStatus  显示服务端连接状态 | 展示服务端主机地址与客户端主机地址 | | exit  退出 | 执行in，out，serverSocket的close()以关闭全部连接 |   **表3.6 SocketClient的其他函数**   1. **User及UserData类**   这两个类的设计相对简单，User类只有username和password两个私有属性，并提供修改及获取这两个属性的所有函数。  UserData类作为仿数据库的类，只有一个Set成员，用来存储通过String表示的各种User数据，并提供一些针对User转String而修改的Set基本方法，如contains，insert。   1. **UDP额外设计**   上述主要算法与设计的1-3都是基于TCP协议，而对于UDP协议的Socket C/S，内容大致不变，但有以下修改：   1. UDP中无论服务端还是客户端，对应的Socket类都改为DatagramSocket 2. UDP中无论服务端还是客户端，增设目的端口私有成员(同时服务端客户端必须指定不同端口才能运行) 3. UDP中不能指定输出输入流的目标，因此采用报文的形式传递信息。因此我自定了UDPUtil类与UDPUtilInterface接口来取代TCP中的out.writeUTF与in.readUTF。   在UDPUtil类中，我基于DatagramSocket与DatagramPacket的send对传出信  息功能(send)进行了封装；基于DatagramPacket对读入信息功能(receive)进行封装。  在UDPUtilInterface接口中，我提供了send和receive简化参数后的抽象标准，以保证对out.writeUTF与in.readUTF进行替换的便利性(分别将四参数与一参数的函数提供一个简化为一参数与零参数的抽象标准，这样参数的数量将与替换前保持一致)。   1. 使UDP中SocketClient与SocketServer类实现UDPUtilInterface的方法，以此来通过最少的修改完成从out.writeUTF与in.readUTF到write与receive的完美替换。 |