武汉纺织大学

Java应用开发课程设计

**基于Socket的聊天程序**

**学 院： 数学与计算机学院**

**班 级： 软件11802**

**姓 名： 成振瑞**

**学 号： 1804240608**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年6月20日**

目 录

[1 需求分析](#_Toc2965)

[1.1用户登陆](#_Toc6465)

[1.2各项功能](#_Toc17655)

[1.2.1登陆验证](#_Toc21322)

[1.2.2 单人聊天](#_Toc30258)

[1.2.3 多人聊天](#_Toc28610)

[1.2.4 退出](#_Toc28851)

[2 系统设计](#_Toc15323)

[2.1用户用例图](#_Toc21899)

[2.2 ER图](#_Toc16671)

[2.3 UML类图（Class Diagram）](#_Toc1640)

[2.4 UML时序图（Sequence Diagram）](#_Toc18606)

[2.4.1 系统登录](#_Toc3306)

[2.5 UML活动图（Activity Diagram）](#_Toc5309)

[2.5.1 登陆](#_Toc12858)

[2.5.1 聊天](#_Toc2967)

[2.5.3 socket模块](#_Toc16882)

[2.5.4 线程模块](#_Toc4032)

[3 系统实现](#_Toc27433)

[3.1 项目结构](#_Toc1061)

[3.2 客户端类Client.java](#_Toc26179)

[3.3 服务器端](#_Toc25901)

[3.3.1 ChatServer.java](#_Toc6856)

[3.3.2 ServerThread.java](#_Toc8757)

[3.3.3 DaoTool.java 12](#_Toc8757)

[3.3.4 UserInfo.java 12](#_Toc8757)

[3.3.5 ChatTools.java 12](#_Toc8757)

[4 系统测试](#_Toc19341)

[5 系统总结](#_Toc30615)

# 1 需求分析

设计一个实时聊天系统，实现“登陆”、“单人聊天”、“多人聊天”、“智能机器人”、 “退出”等功能，具体要求如下：

## 1.1用户登陆

当程序运行时，显示“欢迎使用聊天系统，请登陆”，提示用户依次输入用户名及其密码，当用户名及密码不正确时，显示提示信息“用户名或密码不正确，请重新输入”，**用户最多可以尝试3次，若输入3次均不正确，则给出提示信息“最多只能尝试3次”**，退出系统；

## 1.2各项功能

### 1.3.1 连接服务登陆验证

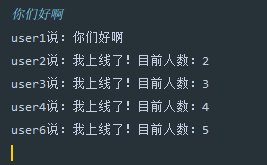


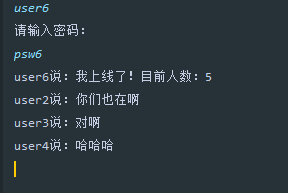
当用户经过登陆验证之后，进入聊天系统。系统会自动提示系统内的其他用户该用户已上线，并显示当前在线人数

之后可通过编辑内容+enter进行聊天

### 1.3.2 多人聊天

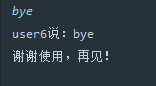
多用户登录时，系统提示当前在线人数，所有用户可在同一聊天室进行聊天





### 1.3.3 退出

当任一用户在聊天框中输入bye的时候，系统会判定，当前用户要退出，并执行退出程序



# 2 系统设计

## 2.1用户用例图

系统分为用户，管理员两个角色，各角色的用户用例如下图2-1所示：

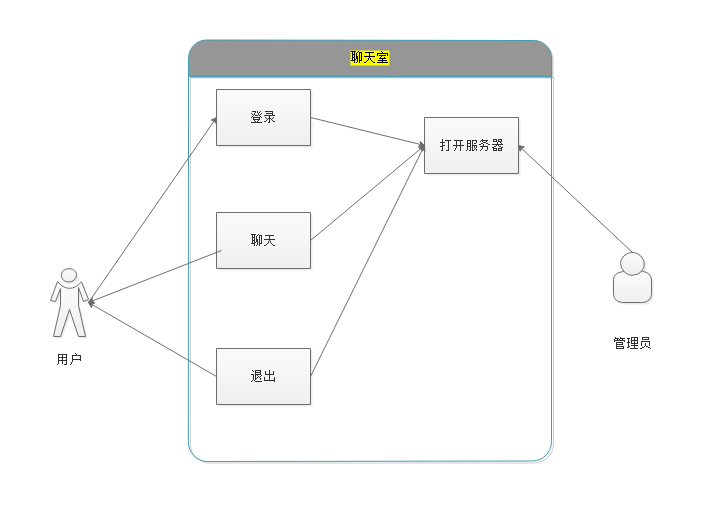
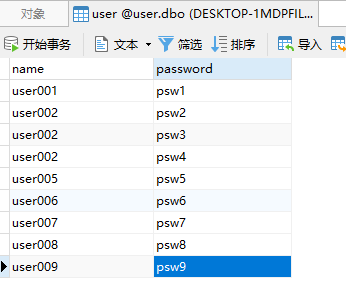


图2-1 用户用例图

## 2.2 ER图

数据库设计1张表，用户表

%

## 2.3 UML类图（Class Diagram）

对于用户功能模块，共设计如下6个类。

* Client类：客户端类，连接服务器，进行客户端输入输出操作。
* ChatServer 类：服务器类，也是主类，里面包含服务器的创建方法 setUpServer (int port) 和主函数入口 main。当程序开始运行时，它会把相应的端口 port 设置为服务器。并让其始终处于待连接状态。每当有客户机连接上来时，就实例化一个线程类（ServerThread）对象，并启动一个线程去处理。（也就相当于我们为每个用户提供了一个独立的线程）。
* ServerThread 类：客户端类。它是一个线程类。里面实现了线程的启动方法 run () 和客户机服务器的通信处理方法 processSocket ()。当然在通信之前我们必须要先验证这个用户信息是否正确。这个验证方法我们在 DaoTool 类中实现。这里直接调用它的验证方法即可
* DaoTool：用户信息验证类。里面实现了用户信息的验证方法 checkLogin ()。并且它还储存了一个模拟的用户信息库 userDB。
* UserInfo：用户信息类。里面保存了每一个用户的信息，包括用户名和密码。定义了获取用户名和密码的方法。
* ChatTools：聊天室类。负责保存当前登录的每一个用户，并且当某一个客户机给服务器发了消息，它需要立即把这条消息转发给其他客户机。

各类的结构及类之间的关系如图2-5所示：

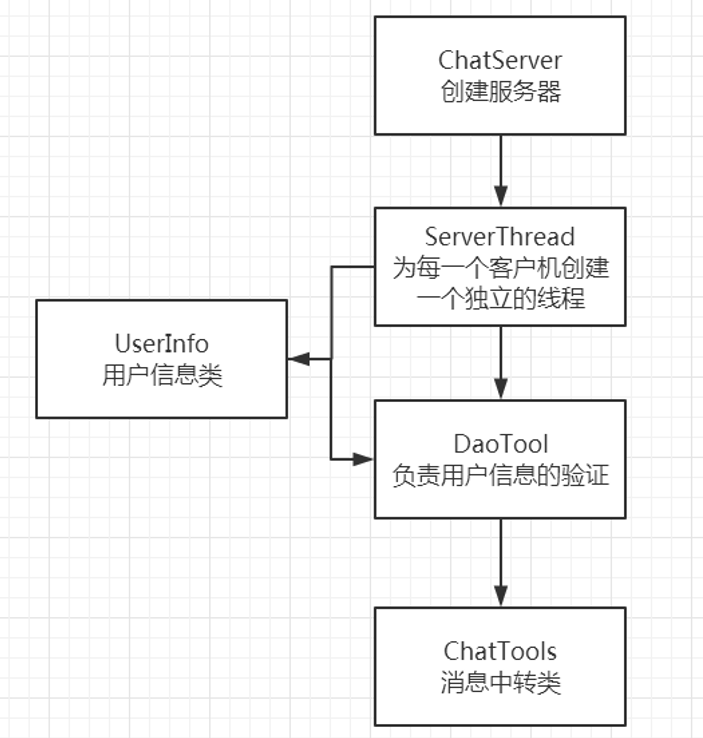
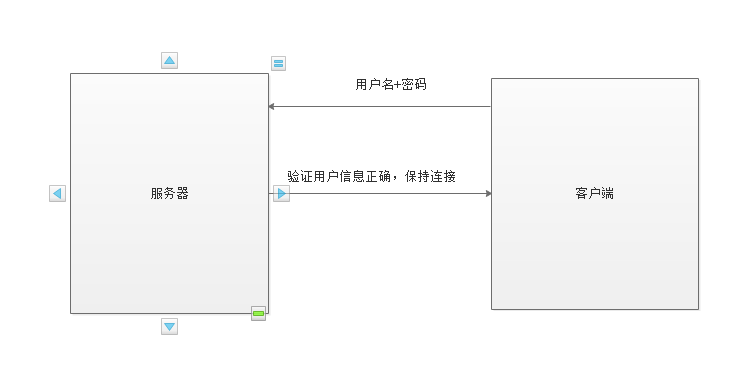


图2-5 用户功能类图

## 2.4 UML时序图（Sequence Diagram）

### 2.4.1 系统登录



## 2.5 UML活动图（Activity Diagram）

### 2.5.1 登陆

登录操作需要验证用户名，密码，只有当用户名和密码同时存在即用户存在的情况下才能成功登录系统，否则会提示错误信息，具体流程如图2-6所示。

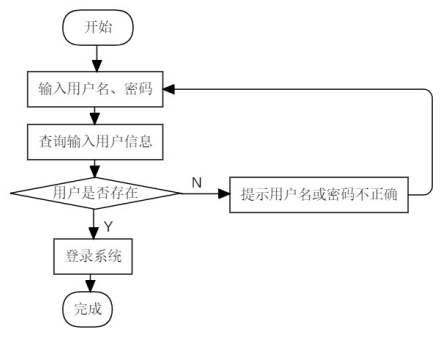
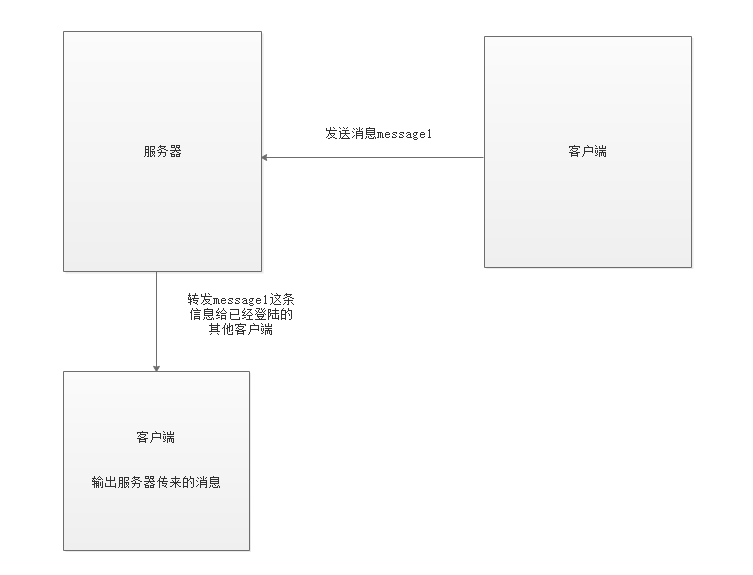
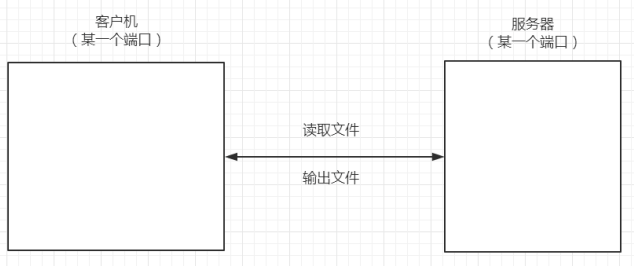


图2-2 登录流程图

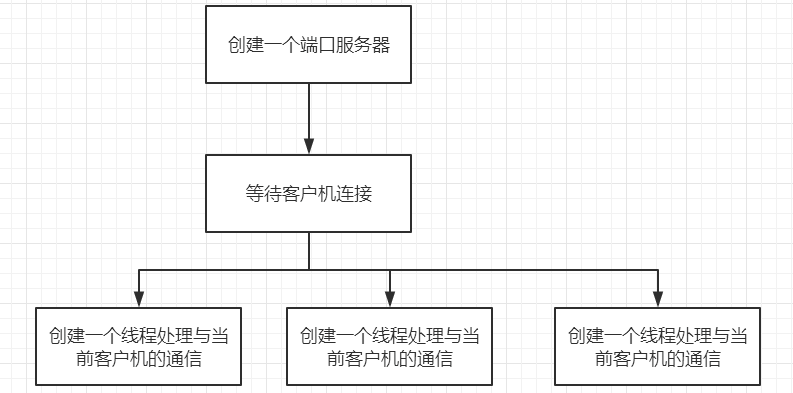
### 2.5.2 聊天



### 2.5.3 socket模块

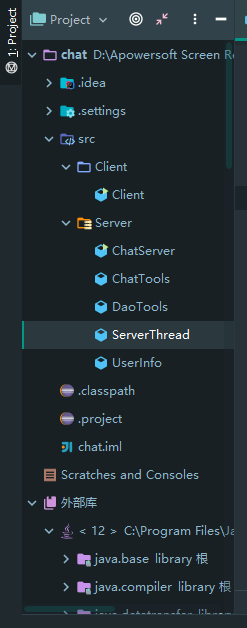


### 2.5.3 多线程模块



# 3 系统实现

## 3.1 项目结构



## 3.2 客户端类Client.java

连接服务器，进行客户端输入输出操作。

*package* Client;  
  
*import* java.io.\*;  
*import* java.net.ServerSocket;  
*import* java.net.Socket;  
*import* java.net.UnknownHostException;  
  
*public class* Client *extends* Thread{  
 InputStream input=*null*;  
 OutputStream output=*null*;  
 BufferedReader bufferinput=*null*;  
 Socket socket=*null*;  
  
 *public* Client(){  
 *//初始化时连接服务器  
 try* {  
 socket=*new* Socket("127.0.0.1",9002);  
 input=socket.getInputStream();  
 output=socket.getOutputStream();  
 *//验证用户信息* login();  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
   
 }  
  
 *public static void* main(String[] args){  
 Client cl=*new* Client();  
 }  
  
 *public void* login(){  
 *try* {  
 *int* Login=0;  
 bufferinput=*new* BufferedReader(*new* InputStreamReader(input));  
 String line;  
 *while*(Login<2){  
 *//获取消息  
 if*((line=bufferinput.readLine())!=*null*){  
 System.*out*.println(line);  
 }  
  
 *//发送消息* BufferedReader brName = *new* BufferedReader(*new* InputStreamReader(System.*in*));  
 String strName;  
 *//控制台在读入数据时并不会自动添加换行符号* strName = brName.readLine()+"\r\n";  
 output.write(strName.getBytes());  
 output.flush();  
 Login++;  
 }  
 chat();  
 } *catch* (UnknownHostException e) {  
 *// TODO Auto-generated catch block* e.printStackTrace();  
 } *catch* (IOException e) {  
 *// TODO Auto-generated catch block* e.printStackTrace();  
 }*finally*{  
 *if*(socket!=*null*){  
 *try* {  
 socket.close();  
 } *catch* (IOException e) {  
 *// TODO Auto-generated catch block* e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 *public void* chat() {  
 start();*//开启一个线程读取当前用户的输入* bufferinput=*new* BufferedReader(*new* InputStreamReader(input));  
 String line;  
 *//获取消息  
 try* {  
 *while*(*true*){  
 *if*((line=bufferinput.readLine())!=*null*){  
 System.*out*.println(line);  
 }  
 }  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }*finally*{  
 *if*(socket!=*null*){  
 *try* {  
 socket.close();  
   
 } *catch* (IOException e) {  
 *// TODO Auto-generated catch block* e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 *public void* run() {  
 *while*(*true*){ *//发送消息* BufferedReader brName = *new* BufferedReader(*new* InputStreamReader(System.*in*));  
 String strName;  
   
 *//控制台在读入数据时并不会自动添加换行符号  
 try* {  
 strName = brName.readLine()+"\r\n";  
 output.write(strName.getBytes());  
 output.flush();  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 }  
   
 }  
}

## 3.3 服务器端

### 3.3.1 ChatServer.java

服务器类，也是主类，里面包含服务器的创建方法 setUpServer (int port) 和主函数入口 main。

*package* Server;  
  
*import* java.io.IOException;  
*import* java.net.ServerSocket;  
*import* java.net.Socket;  
  
*public class* ChatServer {  
 *//主函数入口  
 public static void* main(String[] args) *throws* IOException {  
*//实例化一个服务器类的对象* ChatServer cs=*new* ChatServer();  
*//调用方法，为指定端口创建服务器* cs.setUpServer(9002);  
 }  
  
 *private void* setUpServer(*int* port) *throws* IOException {  
*// TODO Auto-generated method stub* ServerSocket server=*new* ServerSocket(port);  
*//打印出当期创建的服务器端口号* System.*out*.println("服务器创建成功!端口号："+port);  
 *while*(*true*) {  
*//等待连接进入* Socket socket=server.accept();  
 System.*out*.println("进入了一个客户机连接："+socket.getRemoteSocketAddress().toString());  
*//启动一个线程去处理这个对象* ServerThread st=*new* ServerThread(socket);  
 st.start();  
 }  
 }  
}

### 3.3.2 ServerThread.java

它是一个线程类。里面实现了线程的启动方法 run () 和客户机服务器的通信处理方法 processSocket ()。

*package* Server;  
  
*import* java.io.BufferedReader;  
*import* java.io.IOException;  
*import* java.io.InputStream;  
*import* java.io.InputStreamReader;  
*import* java.io.OutputStream;  
*import* java.net.Socket;  
  
*/\*  
 \* 每当有客户机和服务器连接时，都要定义一个接受对象来进行数据的传输  
 \* 从服务器的角度看，这个类就是客户端  
 \*/  
public class* ServerThread *extends* Thread{  
 *private* Socket client;*//线程中的处理对象  
 private* OutputStream ous;*//输出流对象  
 private* UserInfo user;*//用户信息对象  
  
 public* ServerThread(Socket client) {  
 *this*.client=client;  
 }  
  
 *public* UserInfo getOwerUser() {  
 *return this*.user;  
 }  
  
 *public void* run() {  
 *try* {  
 processSocket();  
 } *catch* (IOException e) {  
*// TODO Auto-generated catch block* e.printStackTrace();  
 }  
 }  
  
 *//在显示屏中打印信息  
 public void* sendMsg2Me(String msg) *throws* IOException {  
 msg+="\r\n";  
 ous.write(msg.getBytes());  
 ous.flush();  
 }  
  
  
 *private void* processSocket() *throws* IOException {  
*// TODO Auto-generated method stub* InputStream ins=client.getInputStream();  
 ous=client.getOutputStream();  
 BufferedReader brd=*new* BufferedReader(*new* InputStreamReader(ins));  
 *int* flag = 0;  
 *while* (flag <= 2) {  
 sendMsg2Me("欢迎你来聊天，请输入你的用户名：");  
 String userName=brd.readLine();  
 sendMsg2Me("请输入密码：");  
 String pwd=brd.readLine();  
 user=*new* UserInfo();  
 user.setName(userName);  
 user.setPassword(pwd);  
 *//调用数据库，验证用户是否存在  
 boolean* loginState=DaoTools.*checkLogin*(user);  
 *if*(!loginState) {  
 *//如果不存在这个账号则关闭* flag ++;  
 sendMsg2Me("用户名不存在或者密码错误！你还剩"+ (3-flag) +"次机会！");  
 *//this.closeMe();* }  
 *else* {  
 System.*out*.println("登录成功");  
 *break*;  
 }  
 }  
 *if* (flag == 3) {  
 *this*.closeMe();  
 }  
   
   
 ChatTools.*addClient*(*this*);*//认证成功，把这个用户加入服务器队列* String input=brd.readLine();  
 *while*(!input.equals("bye")) {  
 System.*out*.println("服务器读到的是:"+input);  
 ChatTools.*castMsg*(*this*.user, input);  
 input=brd.readLine();  
 }  
 ChatTools.*castMsg*(*this*.user, "bye");  
 *this*.closeMe();  
 }  
  
 *//关闭当前客户机与服务器的连接。  
 public void* closeMe() *throws* IOException {  
 sendMsg2Me("谢谢使用，再见！");  
 client.close();  
 }  
  
  
}

### 3.3.3 DaoTool.java

用户信息验证类

*package* Server;  
  
*import* java.util.HashMap;  
*import* java.util.*Map*;  
  
*//定义一个处理用户登录信息的类  
public class* DaoTools {  
 *//内存用户信息数据库  
 private static Map*<String,UserInfo>*userDB*=*new* HashMap();  
 *//静态块：模拟生成内存中的用户数据，用户名为1~10  
//当程序启动时这段代码会自动执行向userDB放入数据  
 static* {  
 *for*(*int* i=1;i<=10;i++) {  
 UserInfo user=*new* UserInfo();  
 user.setName("user"+i);  
 user.setPassword("psw"+i);  
 *userDB*.put(user.getName(), user);  
 }  
 }  
  
 *public static boolean* checkLogin(UserInfo user) {  
*//在只验证用户名是否存在  
 if*(*userDB*.containsKey(user.getName())) {  
 *return true*;  
 }  
 System.*out*.println(user.getName()+"用户验证失败！");  
 *return false*;  
 }  
}

### 3.3.4 UserInfo.java

用户信息类

*package* Server;  
  
*//定义一个用户信息的类  
public class* UserInfo {  
 *private* String name;*//用户名  
 private* String password;*//密码  
 private* String loignTime;*//登录时间  
 private* String address;*//客户机端口名  
  
 public* String getName() {  
 *return* name;  
 }  
  
 *public void* setName(String name) {  
*// TODO Auto-generated method stub  
 this*.name=name;  
 }  
  
 *public void* setPassword(String psw) {  
 *this*.password=password;  
 }  
}

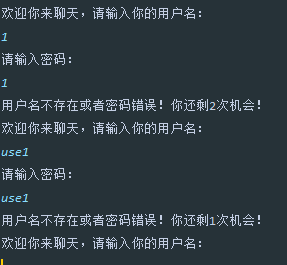
### 3.3.5 ChatTools.java

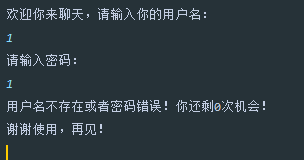
聊天室类

*package* Server;  
  
*import* java.io.IOException;  
*import* java.util.ArrayList;  
  
*/\*  
 \* 定义一个管理类，相当于一个中介，处理线程，转发消息  
 \* 这个只提供方法调用，不需要实例化对象，因此都是静态方法  
 \*/  
public class* ChatTools {  
 *//保存线程处理的对象  
 private static* ArrayList<ServerThread> *stList*=*new* ArrayList();  
 *//不需要实例化类，因此构造器为私有  
 private* ChatTools() {}  
  
 *//将一个客户对应的线程处理对象加入到队列中  
 public static void* addClient(ServerThread st) *throws* IOException {  
 *stList*.add(st);*//将这个线程处理对象加入到队列中  
 castMsg*(st.getOwerUser(),"我上线了！目前人数："+*stList*.size());  
 }  
  
 *//发送消息给其他用户  
 public static void* castMsg(UserInfo sender,String msg) *throws* IOException {  
 msg=sender.getName()+"说："+msg;*//加上说的对象  
 for*(*int* i=0;i<*stList*.size();i++) {  
 ServerThread st=*stList*.get(i);  
 st.sendMsg2Me(msg);*//发消息给每一个客户机* }  
 }  
}

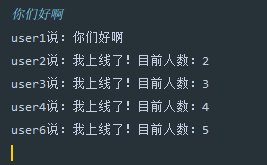
# 4 系统测试

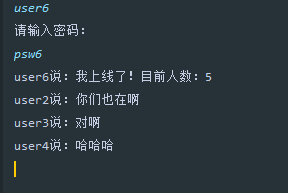


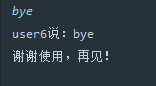












# 5 系统总结

通过socket客户端服务端链接实现建议聊天室，存在的不足：未能添加图形化界面以及未能添加聊天机器人，通过本次Java实践开发，是的我对一些小型项目有了一定的认识以及实践能力，能过通过自己的动手实现一些小型的Java开源项目。