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： 物联网11801**

**姓 名： 马超**

**学 号： 1804240408**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年6月19日**

目 录

[1 需求分析](#_Toc7405)

[1.1用户登陆](#_Toc9628)

[1.2各项功能](#_Toc25433)

[1.2.1 客户端发送消息服务端接收](#_Toc13665)

[1.2.2 服务端发送消息客户端接收](#_Toc4146)

[1.2.3 服务端回复超过30秒调用图灵机器人自动回复 1](#_Toc30960)

[1.2.4 客户端发送文件服务器端接收 1](#_Toc20307)

[1.2.5 服务器端发送文件客户端接收 1](#_Toc26055)

[1.2.6 退出 1](#_Toc13127)

[2 系统设计 1](#_Toc16159)

[2.1 ER图 1](#_Toc15848)

[2.2 UML类图（Class Diagram） 2](#_Toc2575)

[2.3 UML活动图（Activity Diagram） 2](#_Toc18365)

[2.3.1 登陆 2](#_Toc28207)

[3 系统实现 2](#_Toc29097)

[3.1 项目结构 2](#_Toc14899)

[3.2 配置文件 3](#_Toc3956)

[3.2.1 jdbc.properties文件 3](#_Toc31496)

[3.3 客户端Client.java 3](#_Toc5061)

[3.4 单线程服务端SingleServer.java 6](#_Toc30375)

[3.5 服务端SocketServerM.java 11](#_Toc27936)

[4 系统测试 12](#_Toc25887)

[5 系统总结 15](#_Toc15551)

# 1 需求分析

设计一个基于socket的聊天系统：

## 1.1用户登陆

当连接成功后提示输入用户名及密码，登陆成功后即可聊天，若登陆不成功则需重试。

## 1.2各项功能

### 1.2.1 客户端发送消息服务端接收

客户端发出消息服务端即可接收到消息。

### 1.2.2 服务端发送消息客户端接收

服务端发出消息客户端即可接收到消息。

### 1.2.3 服务端回复超过30秒调用图灵机器人自动回复

若服务端超过30秒未回复则由图灵机器人自动回复。

### 1.2.4 客户端发送文件服务器端接收

客户端通过send +路径的方式发送文件，服务端填写接受路径后即可接收。

### 1.2.5 服务端发送文件客户端接收

服务端通过send +路径的方式发送文件，客户端填写接受路径后即可接收。

### 1.2.6 退出

客户端输入bye双端退出。

# 2 系统设计

## 2.1 ER图

数据库设计1张表，用户表，对应的ER图如图3-1所示。

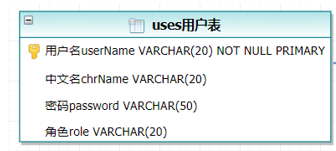


图3-1 数据库ER图

## 2.2 UML类图（Class Diagram）

对于用户功能模块，共设计如下6个类。

* 单个服务类SingleServer：连接数据库并获取连接对象，关闭连接对象。

## 2.3 UML活动图（Activity Diagram）

### 2.3.1 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2-2所示。

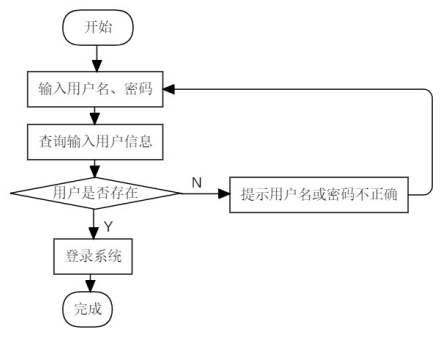
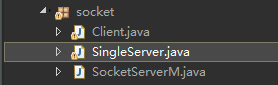
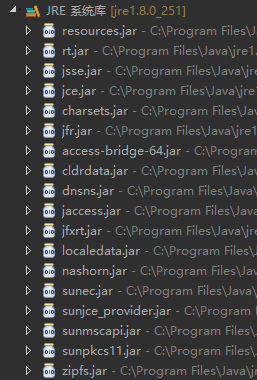


图2-2 登录流程图

# 3 系统实现

## 3.1 项目结构





## 3.2 配置文件

在项目下新建Source Folder，名为resources

### 3.2.1 jdbc.properties文件

该配置文件主要是以文件形式保存数据库的驱动类名称，连接数据库的URL地址，访问数据库的用户名及对应的密码，程序运行时会读取该文件相关信息，避免硬编码，当相关信息发生变化时，只需修改配置文件而不用修改源代码，增加程序的可扩展型。

DBDRIVER=com.mysql.jdbc.Driver

DBURL=jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf8&useSSL=false

DBUSER=root

PASSWORD=000000

## 3.3 客户端Client.java

|  |
| --- |
| package socket;  import java.io.BufferedReader;  import java.io.DataOutputStream;  import java.io.File;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.io.OutputStream;  import java.io.OutputStreamWriter;  import java.io.PrintWriter;  import java.net.Socket;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.Scanner;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class Client {  public static void main(String[] args) throws Exception{  int port = 12000;  String host = "127.0.0.1";  Socket socket = new Socket(host,port);    InputStream is = socket.getInputStream();  OutputStream os = socket.getOutputStream();    BufferedReader br = new BufferedReader(new InputStreamReader(is));  PrintWriter pw = new PrintWriter(new OutputStreamWriter(os));    Scanner scan = new Scanner(System.***in***);  pw.println("connect");  pw.flush();    String ans= br.readLine();  System.***out***.println(ans);    while(!ans.equals("登陆成功")) {    ans= br.readLine();  System.***out***.println(ans);    String user = scan.nextLine();  pw.println(user);  pw.flush();    ans= br.readLine();  System.***out***.println(ans);    String pass = scan.nextLine();  pw.println(pass);  pw.flush();    ans= br.readLine();  System.***out***.println(ans);  }    while(true) {  System.***out***.print("请输入要发送的信息：");  String str = scan.nextLine();  pw.println(str);  pw.flush();    Pattern pt = Pattern.*compile*("send.+");  Matcher mt = pt.matcher(str);  while(mt.find()) {  String st = mt.group();  st = st.replaceAll("send ","");  *send*(st,os);  }    String aws = br.readLine();  Pattern ptt = Pattern.*compile*("send.+");  Matcher mtt = pt.matcher(aws);  if(mtt.find()) {  *receive*(is);  }  else  System.***out***.println("回复："+aws);      if(str.equals("bye"))  break;  }  socket.close();  System.***out***.println("客户端程序结束。");  }    public static void send(String str, OutputStream os) throws IOException {    FileInputStream fis = new FileInputStream(str);  byte[] bytes = new byte[1024];  int data;  while((data = fis.read(bytes))!= -1){  os.write(bytes, 0, data);  }  fis.close();  System.***out***.println("发送成功!");  }    public static void receive(InputStream is) throws Exception {  Scanner scan = new Scanner(System.***in***);  System.***out***.print("请输入保存路径：");  String path = scan.nextLine();    FileOutputStream fos = new FileOutputStream(path);  byte[] bytes = new byte[1024];  int data;  while((data = is.read(bytes))!=-1){  fos.write(bytes, 0, data);  }  fos.close();  System.***out***.println("接收成功");  }    } |

## 3.4 单线程服务端SingleServer.java

|  |
| --- |
| package socket;  import java.io.BufferedReader;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.io.OutputStream;  import java.io.OutputStreamWriter;  import java.io.PrintWriter;  import java.net.HttpURLConnection;  import java.net.Socket;  import java.net.URL;  import java.net.URLEncoder;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.Scanner;  import java.util.concurrent.Callable;  import java.util.concurrent.ExecutionException;  import java.util.concurrent.ExecutorService;  import java.util.concurrent.Executors;  import java.util.concurrent.Future;  import java.util.concurrent.TimeUnit;  import java.util.concurrent.TimeoutException;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import com.google.gson.JsonArray;  import com.google.gson.JsonElement;  import com.google.gson.JsonObject;  import com.google.gson.JsonParser;  public class SingleServer implements Runnable{  private Socket socket;  private int clientNo;    public SingleServer(Socket socket, int clientNo) {  this.socket = socket;  this.clientNo = clientNo;  }    public void run() {  try {  InputStream is = socket.getInputStream();  OutputStream os = socket.getOutputStream();    BufferedReader br = new BufferedReader(new InputStreamReader(is));  PrintWriter pw = new PrintWriter(new OutputStreamWriter(os));    String tt = br.readLine();  System.***out***.println(tt);    pw.println("连接服务器成功！\n请输入用户名:");  pw.flush();  String userName = br.readLine();  System.***out***.println(userName);    pw.println("请输入密码:");  pw.flush();  String password = br.readLine();  System.***out***.println(password);    while(!*login*(userName,password)) {  pw.println("用户名或密码错误！\n请重新输入用户名:");  pw.flush();  userName = br.readLine();    pw.println("请输入密码:");  pw.flush();  password = br.readLine();  }    pw.println("登陆成功");  pw.flush();    while(true) {  String Str = br.readLine();    Pattern pt = Pattern.*compile*("send.+");  Matcher mt = pt.matcher(Str);  if(mt.find()) {  *receive*(is);  System.***out***.print("接收成功");  pw.println("接收成功");  pw.flush();  }else {    System.***out***.println("客户端来信："+Str);    if(Str.equalsIgnoreCase("bye")) {  pw.println("bye");  pw.flush();  break;  }    String aws = *answer*();    if(aws.equalsIgnoreCase("#回复超时#")) {  String INFO = URLEncoder.*encode*(Str, "utf-8");  String getURL = "http://www.tuling123.com/openapi/api?key=1c99470a8a8354e248a4c229234d14af"+ "&info=" + INFO;  URL getUrl = new URL(getURL);  HttpURLConnection connection = (HttpURLConnection) getUrl.openConnection();  connection.connect();  BufferedReader reader = new BufferedReader(new InputStreamReader( connection.getInputStream(), "utf-8"));  String line = reader.readLine();    JsonParser parser = new ~~JsonParser~~();  JsonElement element = parser.~~parse~~(line);  if (element.isJsonObject()) {  JsonObject object = element.getAsJsonObject(); // 转化为对象  pw.println("自动回复："+object.get("text").getAsString());  pw.flush();  }  reader.close();  }  else {  pw.println(aws);  pw.flush();    Pattern ptt = Pattern.*compile*("send.+");  Matcher mtt = pt.matcher(aws);  if(mtt.find()) {  String st = mtt.group();  st = st.replaceAll("send ","");  *send*(st,os);  }  }  }      };  }catch(Exception e) {  e.printStackTrace();  }finally {  System.***out***.println("与客户端" + clientNo + "通信结束");  try {  socket.close();  }catch(IOException e) {  e.printStackTrace();  }  }  }    public static boolean login(String userName, String password) throws Exception {  Scanner scan = new Scanner(System.***in***);  Class.*forName*("com.mysql.jdbc.Driver");    Connection con = DriverManager.*getConnection*("jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf-8&useSSL=false"  ,"root"  ,"000000");  //连接的数据库  String sql = "SELECT \* FROM user WHERE username = ? AND password = ?"; //数据库查询语句  PreparedStatement pst = con.prepareStatement(sql);  pst.setString(1, userName);  pst.setString(2, password);  ResultSet rs = pst.executeQuery();  boolean res = rs.next();  con.close();    return res;  }    public static String answer() throws InterruptedException,ExecutionException {  final ExecutorService exec = Executors.*newFixedThreadPool*(1);    Callable<String> call = new Callable<String>() {  public String call() throws Exception {  Scanner scan = new Scanner(System.***in***);  System.***out***.print("您的回复：");  String ans = scan.nextLine();  return ans;  }  };  try {  Future<String> future = exec.submit(call);  String obj = future.get(1000 \* 30, *TimeUnit*.***MILLISECONDS***); //任务处理超时时间设为 30 秒  return obj;  } catch (TimeoutException ex) {  System.***out***.println("\n回复超时");  return "#回复超时#";  } catch (Exception e) {  System.***out***.println("回复失败");  return "#回复失败#";  }  // 关闭线程池    }    public static void send(String str, OutputStream os) throws IOException {    FileInputStream fis = new FileInputStream(str);  byte[] bytes = new byte[1024];  int data;  while((data = fis.read(bytes))!= -1){  os.write(bytes, 0, data);  }  fis.close();  System.***out***.println("发送成功!");  }  public static void receive(InputStream is) throws Exception {  Scanner scan = new Scanner(System.***in***);  System.***out***.print("请输入保存路径：");  String path = scan.nextLine();    FileOutputStream fos = new FileOutputStream(path);  byte[] bytes = new byte[1024];  int data;  while((data = is.read(bytes))!=-1){  fos.write(bytes, 0, data);  }  fos.close();    }  } |

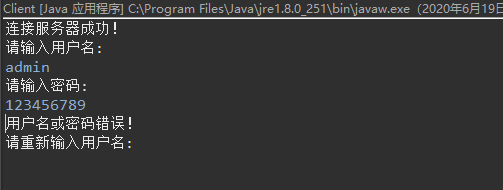
## 3.5 服务端SocketServerM.java

|  |
| --- |
| package socket;  import java.io.IOException;  import java.net.ServerSocket;  import java.net.Socket;  public class SocketServerM {  public static void main(String[] args) throws IOException {  int port = 12000;  int clientNo = 1;  ServerSocket serverSocket = new ServerSocket(port);  try {  while(true) {  Socket socket = serverSocket.accept();  SingleServer singleServer = new SingleServer(socket,clientNo);  Thread thread = new Thread(singleServer);  thread.start();  clientNo++;  }  }finally {  serverSocket.close();  }  }  } |

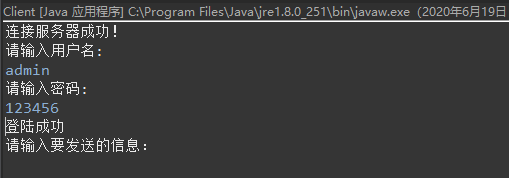
# 4 系统测试

登录模块：

登陆失败演示：

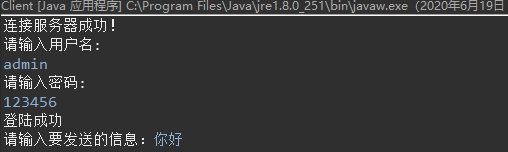


登陆成功演示：

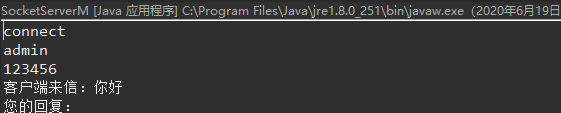


客户端向服务端发送信息演示：

客户端：



服务端：



服务端回复客户端消息：

客户端：



服务端：



客户端发送文件服务端接收：

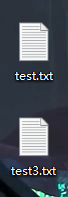
客户端：

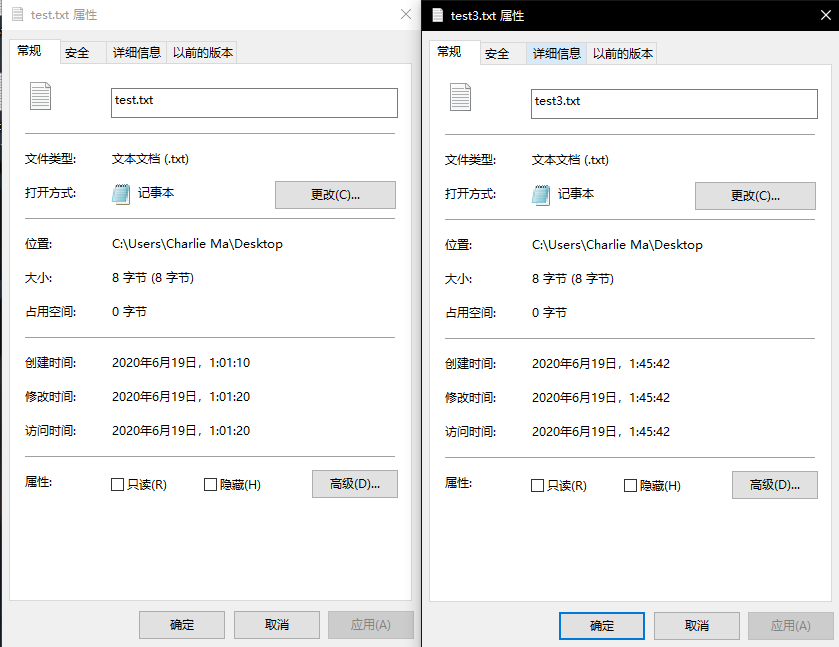


服务端：



传输结果：





服务端发送文件客户端接收：

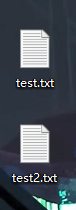
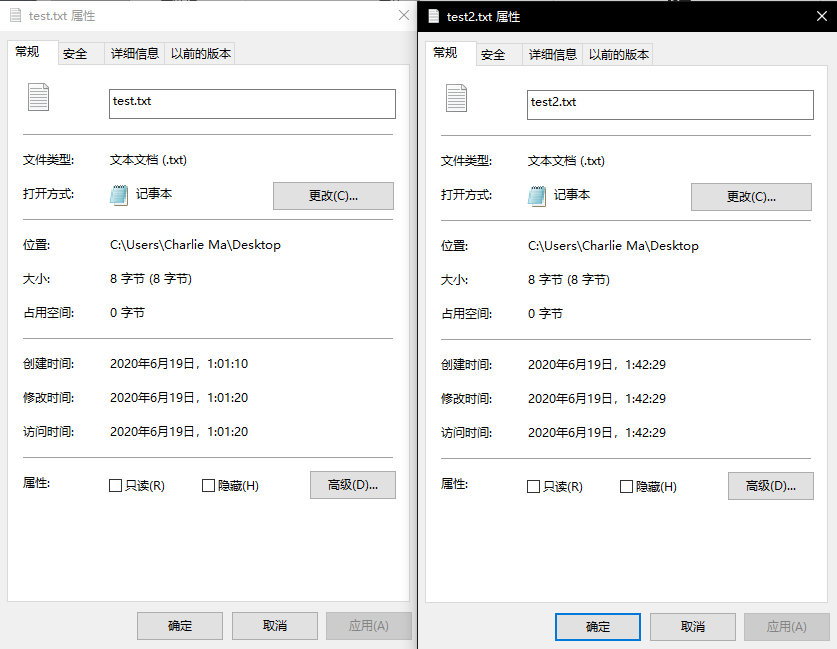
客户端：



服务端：



传输结果：

服务端回复超过30秒调用图灵机器人自动回复：

客户端：



服务端：



# 5 系统总结

这次作业中主要解决了socket传输，API的引入，程序超时检测等问题，这些问题主要通过观看老师的教案以及一些样例程序解决，系统内不足：客户端与服务端只能一人一句，通过本门可学习了Java语言、网络编程及一些今后用到的编程知识，如减少硬编码的方法等，建议本门课可以多一些习题的讲解。今后可以使用java开发一些简单的程序，本次作业所有功能均已实现，但存在一些不足，故自评成绩为80分。