重庆大学大数据与软件学院

上 机 实 验 报 告

上机实践项目	ATM 系统
课程名称	计算机网络

姓名	冯俊杰，王文博	成绩	
学号	20221120，20221497	教师	**胡海波**
班级	软件工程 4 班	日期	2024-5-10

《计算机网络》上机实验报告

开课实验室：　　年　月　日

姓　名	冯俊杰	年级、班级	2022 级软件工程 4 班	学号	**20221120**
上机（项目）名称		ATM 系统		指导教师	胡海波
教师评语					

教师签名：

年　月　日

一、上机目的

本实验的目的是让学生通过实践操作，深入理解计算机网络中的 Socket 编程原理，并掌握如何使用 Socket 编程来实现客户端和服务器端之间的通信。通过设计并实现一个 ATM 系统，学生能够将理论知识与实际应用相结合，提高编程能力，增强解决实际问题的能力。此外，实验还旨在培养学生的团队协作精神，通过小组合作完成项目，学习如何在团队中分工合作，共同解决问题。

二、基本原理

本实验基于 TCP/IP 协议栈，使用 Socket 编程来实现客户端和服务器端之间的通信。Socket 是网络通信的端点，可以看作是两个网络应用程序之间通信的桥梁。在 TCP 协议下，Socket 提供了一个可靠的、面向连接的通信通道。

1. **TCP Socket 编程**：TCP（Transmission Control Protocol，传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议。Socket 编程使用 TCP 协议来确保数据的可靠传输。
2. **客户端-服务器模型**：在本实验中，ATM 系统采用客户端-服务器模型。服务器端持续监听来自客户端的连接请求，一旦客户端发起连接，服务器端接受连接并建立通信通道。
3. **数据交换格式**：根据 RFC-20222022 协议，客户端和服务器端需要按照规定的格式交换数据。例如，客户端发送的请求和服务器端返回的响应都需要遵循特定的格式。
4. **异常处理与日志记录**：服务器端需要能够处理各种异常情况，并对操作进行日志记录，以便于问题的追踪和系统的维护。

三、使用的软件、硬件

软件：idea 集成环境，mysql 数据库，windows 系统

硬件：PC

四、上机操作步骤

- Homework 2
    - 结合 1-2 章所学知识 (协议、TCP Socket Programming)
    - 用你们自己喜欢的一种程序语言，但一定要遵照 RFC-20222022（具体协议见 https://shimo.im/docs/d1hLMvSAfjJ7uq9l ）
    - 设计开发 ATM 客户端、服务器端
        - 客户端：需要 GUI
        - 服务器端：不需要 GUI，但需要读取数据文件、记录日志；默认端口号 2525
    - 2 个人 1 组：68 位同学分为 34 个组
        - https://shimo.im/sheets/fIz1BfyOyMxFURSa/MODOC
        - 组内的 C-S，先要编码并测试通过
    - 第 2 次实验课进行小组之间的测试
        - 客户端写一个完整的测试用例，访问其他组的服务器端
        - 服务器端要对所有的异常和取款记录日志
        - 程序代码+文档+报告：Github 链接
    - DDL1: 第 7 周周四实验课，进行交叉测试
    - DDL2: tba，提交代码和文档、测试结果

五、过程原始记录(数据、图表、计算等)

一：用户端

（1） 功能设计：

本系统根据 RFC20222022 协议设计，在用户端主要能够实现账户登录，存款，取款，余额查询，退出账户等功能。

Client **Server**

HELO <userid> ----------> (check if valid userid)

<---------- 500 sp AUTH REQUIRED!

PASS <passwd> ----------> (check password)

<---------- 525 OK! (//password is OK)

BALA ----------> (check balance from database)

<---------- AMNT:<amnt>

WDRA <amnt> ----------> (check if enough money to cover withdrawal)

<---------- 525 OK (if enough, update database)

(ATM dispenses)

<---------- 401 sp ERROR! (else)

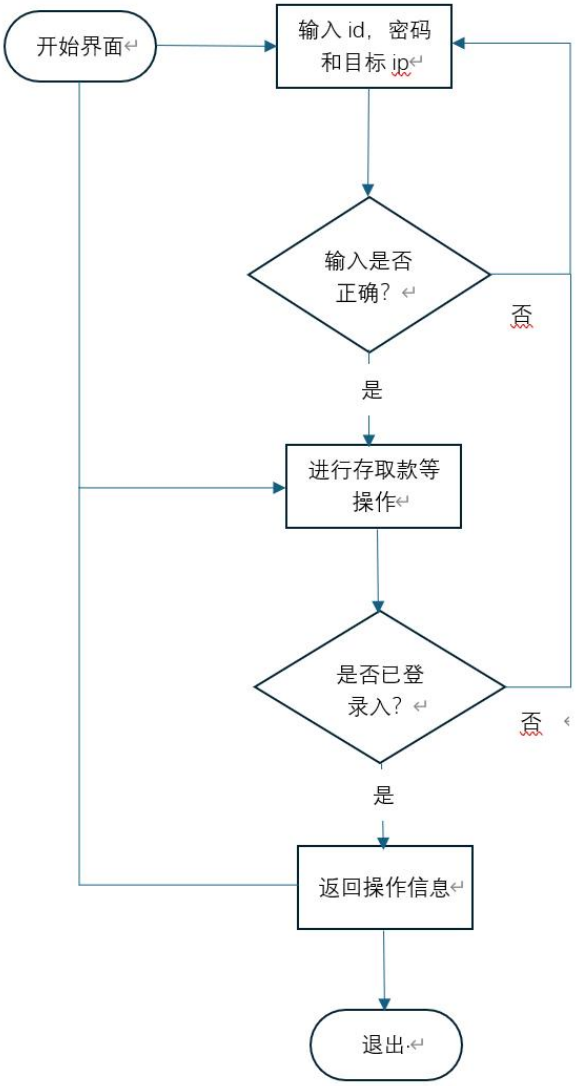
BYE ---------->

<---------- BYE

1. Messages from ATM to Server

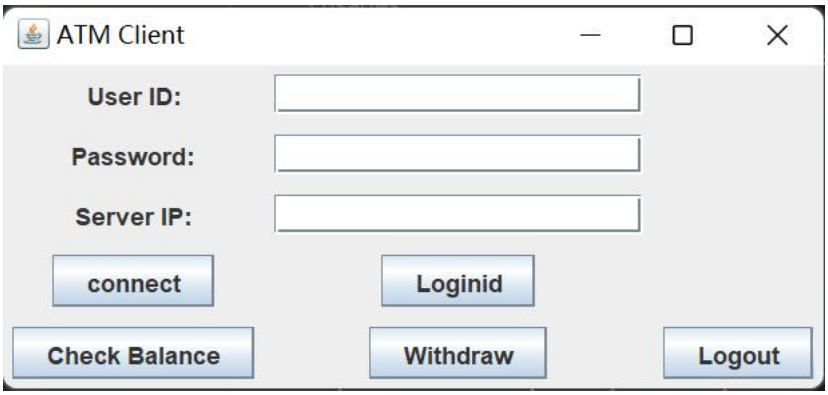
Msg name	Purpose
HELO sp <userid>	Let server know that there is a card in the ATM machine
	ATM transmits user ID (cardNo.) to Server
PASS sp <passwd>	User enters PIN (password), which is sent to server
BALA	User requests balance
WDRA sp <amount>	User asks to withdraw money
BYE	user all done

流程图：

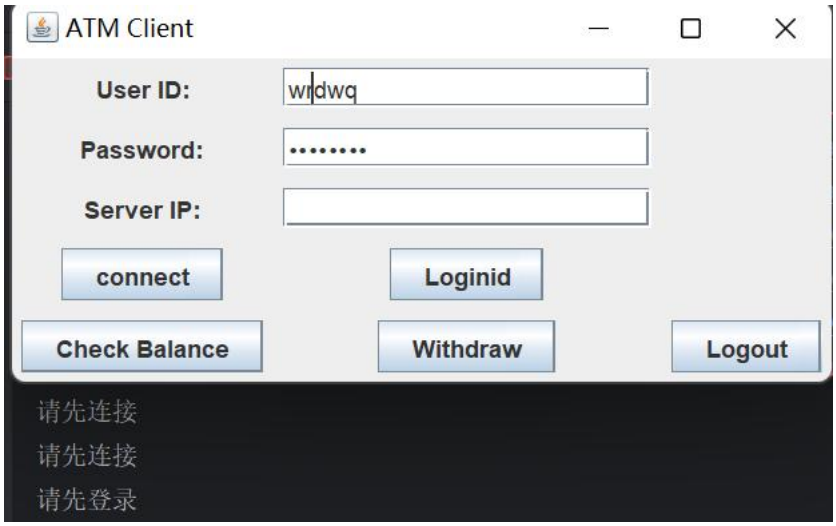


（2）使用：

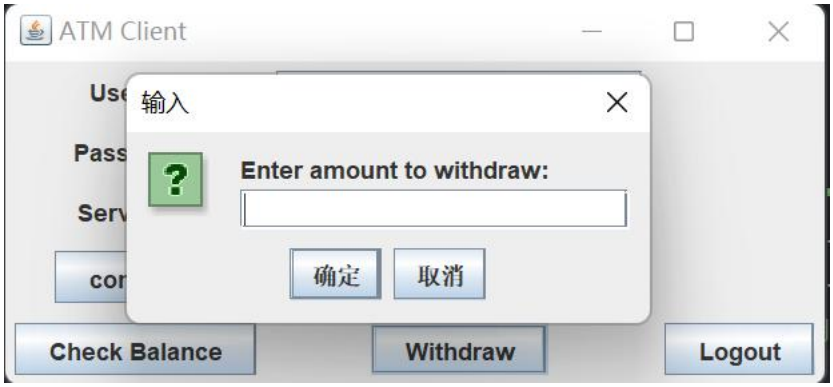
点击运行后，会弹出如下窗口：



在运行中，可同时输入用户 id，用户密码以及服务器 ip，并配置了错误检测机制。例如，若仅仅输入了 id 和密码而未输入 ip 地址点击联机后，系统会提示“未输入 ip”报错等以保证系统完整性。



若登录成功，则可以成功进行存取款操作等，点击 withdraw 输入取款金额，则可以成功取款。若输入有误，则会报错。



二：服务器端

在服务器端，主要能实现用户端响应，非法输入检测，数据库访问等功能。同时也能记录每个用户对应的操作记录日志。

服务器运行记录，用以记录服务器的所有响应记录：

```
客户端已连接：10.234.108.52  2024-05-25T18:49:23.265
连接成功
获取成功
客户端消息：HELO stu007
服务端消息：500 AUTH REQUIRE
客户端消息：PASS ia
服务端消息：401 ERROR!
客户端消息：HELO stu007
服务端消息：500 AUTH REQUIRE
客户端消息：PASS iam007
服务端消息：525 OK!
客户端消息：WDRA 4968y73u89
服务端消息：401 ERROR!
客户端消息：WDRA 9430qngfd
服务端消息：401 ERROR!
客户端消息：WDRA rtgshsh
服务端消息：401 ERROR!
客户端消息：WDRA 356@#$%^&*
服务端消息：401 ERROR!
客户端消息：BYE
服务端消息：BYE
客户端连接已断开  2024-05-25T18:49:56.099
```

用户表：

ID	passwd	balance
2022149720...	123456	499390
stu007	iam007	236034.99948998546
NULL	NULL	NULL

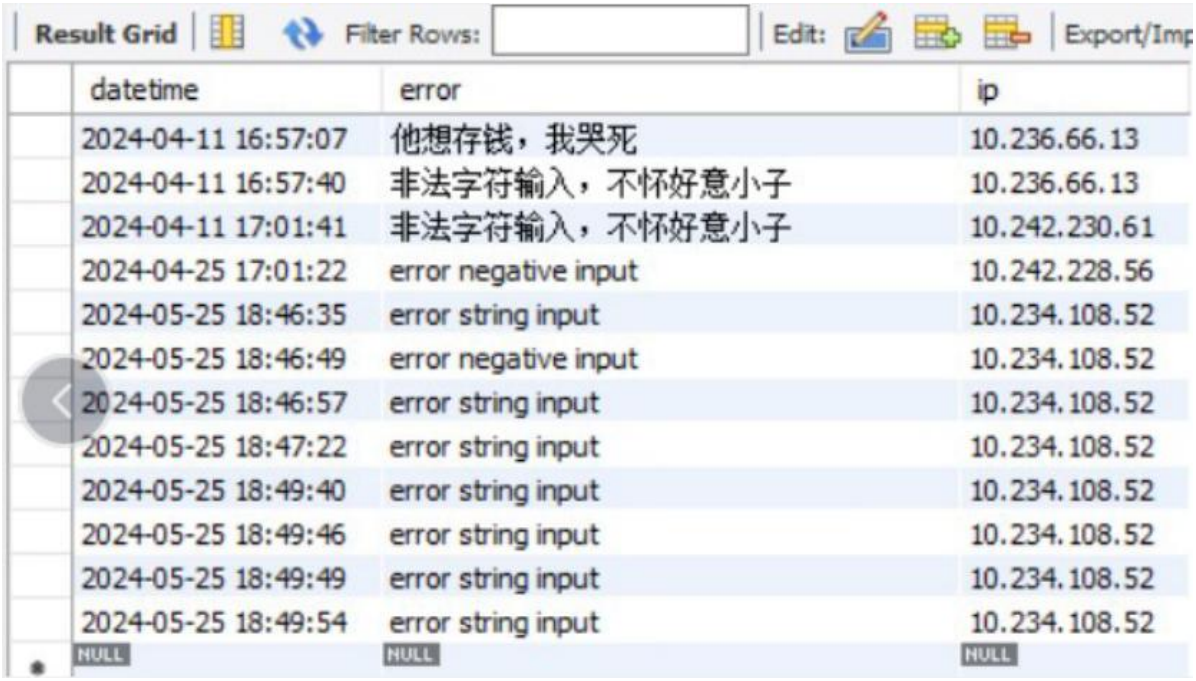
用户存款记录：

Result Grid | Filter Rows: | Edit: | Expo

NO	ID	money	datetime
482	stu007	100	2024-04-11 16:37:33
483	stu007	100	2024-04-11 16:37:33
484	stu007	0.0001	2024-04-11 16:45:25
485	stu007	0.0001	2024-04-11 16:45:41
486	stu007	100	2024-04-11 16:46:31
487	stu007	1e-25	2024-04-11 16:47:25
488	stu007	0.00001	2024-04-11 16:47:44
489	stu007	0.0003	2024-04-11 16:47:58
490	stu007	0.000...	2024-04-11 16:48:39
491	2022149720221120	100	2024-04-25 17:01:18
492	2022149720221120	500	2024-04-25 17:37:01
493	stu007	4000	2024-05-25 18:45:55
NULL	NULL	NULL	NULL

record_money 2

错误检测日志报告，用以记录非法字符输入：



Result Grid | Filter Rows: | Edit: | Export/Imp

datetime	error	ip
2024-04-11 16:57:07	他想存钱，我哭死	10.236.66.13
2024-04-11 16:57:40	非法字符输入，不怀好意小子	10.236.66.13
2024-04-11 17:01:41	非法字符输入，不怀好意小子	10.242.230.61
2024-04-25 17:01:22	error negative input	10.242.228.56
2024-05-25 18:46:35	error string input	10.234.108.52
2024-05-25 18:46:49	error negative input	10.234.108.52
2024-05-25 18:46:57	error string input	10.234.108.52
2024-05-25 18:47:22	error string input	10.234.108.52
2024-05-25 18:49:40	error string input	10.234.108.52
2024-05-25 18:49:46	error string input	10.234.108.52
2024-05-25 18:49:49	error string input	10.234.108.52
2024-05-25 18:49:54	error string input	10.234.108.52
NULL	NULL	NULL

三：源代码

(1):用户端

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;

public class ATMClient_2 extends JFrame implements ActionListener {
    private JTextField userIdField;
    private JPasswordField passwordField;
    private JButton loginButton_id;
    private JButton connectButton;
    private String ip;
    private JTextField iptextFiled;
    private JButton checkBalanceButton;
    private JButton withdrawButton;
    private JButton logoutButton;
    private Socket socket;
    private PrintWriter out;
    private BufferedReader in;
    private boolean connected = false;
    private boolean loggedIn = false; // 登录状态标记



    public ATMClient_2() {
        super("ATM Client");

        // 创建界面组件
        userIdField = new JTextField(20);
```

```
        passwordField = new JPasswordField(20);
        iptextFiled = new JTextField(20);
        loginButton_id = new JButton("Loginid");
        connectButton = new JButton("connect");
        checkBalanceButton = new JButton("Check Balance");
        withdrawButton = new JButton("Withdraw");
        logoutButton = new JButton("Logout");

        // 设置布局面板
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);

        // 添加用户名和密码输入框及登录按钮
        gbc.gridx = 0;
        gbc.gridy = 0;
        add(new JLabel("User ID:"), gbc);
        gbc.gridx = 1;
        add(userIdField, gbc);
        gbc.gridx = 0;
        gbc.gridy = 1;
        add(new JLabel("Password:"), gbc);
        gbc.gridx = 1;
        add(passwordField, gbc);
        gbc.gridx = 0;
        gbc.gridy = 2;
        add(new JLabel("Server IP:"), gbc);
        gbc.gridx = 1;
        add(iptextFiled, gbc);
        gbc.gridx = 1;
        gbc.gridy = 3;
        add(loginButton_id, gbc);

        gbc.gridx = 0;
        gbc.gridy = 3;
        add(connectButton,gbc);

        // 添加操作按钮
        gbc.gridx = 0;
        gbc.gridy = 4;
        add(checkBalanceButton, gbc);
        gbc.gridx = 1;
        add(withdrawButton, gbc);
        gbc.gridx = 2;
        add(logoutButton, gbc);

        // 设置窗口属性
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
        setVisible(true);

        loginButton_id.addActionListener(this);
        connectButton.addActionListener(this);
        checkBalanceButton.addActionListener(this);
        withdrawButton.addActionListener(this);
        logoutButton.addActionListener(this);

    }
```

```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == loginButton_id) {
            if(connected == true){
                String userId = userIdField.getText();
                char[] passwordChars = passwordField.getPassword();
                String password = new String(passwordChars);

                // 发送登录请求给服务器
                sendMessage("HELO " + userId);
                sendMessage("PASS "+password);
            }else{
                System.out.println("请先连接");
            }


        } else if(e.getSource() == connectButton){
            //建立 socket 连接
            ip = iptextFiled.getText();
            try{
                socket = new Socket(ip, 2525);
                //建立输出流向服务端发送消息
                out = new PrintWriter(socket.getOutputStream(), true);
                //输入流接受消息
                in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                connected = true;
                System.out.println("连接成功");
            }catch(IOException esm){
                esm.printStackTrace();
            }
        } else if (e.getSource() == checkBalanceButton ) {
            //检查余额
            if(loggedIn == true){
                sendMessage("BALA");
            }else{
                System.out.println("请先登录");
            }
            // 发送查询余额请求给服务器

        } else if (e.getSource() == withdrawButton) {
            //取钱
            if(loggedIn==true){
                String amount = JOptionPane.showInputDialog(this, "Enter amount
to withdraw:");
                sendMessage("WDRA " + amount);
            }else{
                System.out.println("请先登录");
            }

        } else if (e.getSource() == logoutButton && loggedIn == true) {
            // 发送退出请求给服务器
            sendMessage("BYE");
            try {
                // 关闭套接字和输入输出流
                in.close();
                out.close();
                socket.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
```

```
            System.exit(0); // 退出客户端程序
        }
    }

    private void sendMessage(String message) {
        out.println(message);

        try {
            String response = in.readLine();

            if(message.startsWith("BAL")){
                System.out.println("当前余额为: " + response.substring(5));
            }else if(message.startsWith("HE")){
                if(response.startsWith("401")){
                    System.out.println("密码或用户名错误");
                }

            }else if(message.startsWith("PASS")){
                if(response.startsWith("401")){
                    System.out.println("密码或用户名错误");
                }else if(response.startsWith("525")){
                    System.out.println("登录成功");
                    loggedIn = true;
                }
            }else if(message.startsWith("WD")){
                if(response.startsWith("401")){
                    System.out.println("余额不足或输入有误");
                }
                else if(response.startsWith("AMNT")){
                    System.out.println("取钱成功");
                }
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new ATMClient_2());

    }
}
```

（2）：服务器

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.time.LocalDateTime;

import static jdk.nashorn.internal.objects.NativeString.substr;
import static jdk.nashorn.internal.objects.NativeString.substring;
```

```
public class Server {
    public static void main(String[] args) {
        try {
            // 创建 ServerSocket，监听指定端口
            ServerSocket serverSocket = new ServerSocket(2525);

            System.out.println("服务器已启动，等待客户端连接...");

            // 循环接收客户端连接
            while (true) {
                // 等待客户端连接
                Socket clientSocket = serverSocket.accept();

                LocalDateTime dateTime = LocalDateTime.now();
                System.out.println("客户端已连接： " +
clientSocket.getInetAddress().getHostAddress()+"  "+dateTime);
                // 创建新线程或处理器处理客户端通信
                ClientHandler handler = new ClientHandler(clientSocket);
                handler.start(); // 启动处理器线程
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

// 客户端处理器类，用于处理单个客户端的通信
class ClientHandler extends Thread {
    private Socket clientSocket;

    public ClientHandler(Socket socket) {
        this.clientSocket = socket;
    }

    @Override
    public void run() {
        Connection conn=null;
        PreparedStatement stmt;
        //监视操作次数
        int operate_times=0;
        //Statement sql;
        ResultSet rs;
        String url = "jdbc:mysql://127.0.0.1:3306/atm_sever";
        String username = "root";
        String password = "superhunstein";

        try {
            //连接数据库
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("连接成功");
        }catch(Exception f){
            System.out.println("数据库连接失败");
        }
        try{
            //获取链接
            conn = DriverManager.getConnection(url, username, password);
            System.out.println("获取成功");
        }catch(Exception f){
```

```
            System.out.println("链接获取失败");
        }

        try {
            // 获取输入流，用于接收客户端发送的数据
            BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            DataOutputStream  outToClient = new
DataOutputStream(clientSocket.getOutputStream());

            // 读取客户端发送的数据
            String clientData;
            String userID = null;
            String passwd;
            while ((clientData = input.readLine()) != null) {
                if(operate_times>20){
                    //记录异常操作
                    LocalDateTime dateTime = LocalDateTime.now();
                    if (conn != null) {
                        stmt = conn.prepareStatement("insert into error_logs
values('" + dateTime + "',?,?)");
                        stmt.setString(1, "error frequent operate!");
                        stmt.setString(2,
clientSocket.getInetAddress().getHostAddress());
                        int ok_insert = stmt.executeUpdate();
                    }
                    outToClient.writeBytes(("error
operate??????????????????????????????\n"));
                    outToClient.flush();
                    // 关闭流和套接字
                    input.close();
                    outToClient.close();
                    clientSocket.close();


                    System.out.println("客户端连接已断开  " + dateTime + "\n\n\n");

                }
                System.out.println("客户端消息: " + clientData);
                String headData=substr(clientData,0,3);
                switch(headData){
                    case "HEL":
                        operate_times++;
                        userID=substring(clientData,5);
                        outToClient.writeBytes("500 AUTH REQUIRE\n");
                        System.out.println("服务端消息: 500 AUTH REQUIRE");
                        outToClient.flush();
                        break;
                    case "PAS":
                        operate_times++;
                        passwd=substring(clientData,5);
                        try{
                            if(conn!=null){
                                stmt=conn.prepareStatement("select * from info
where ID=? and passwd=?");
                                stmt.setString(1,userID);
                                stmt.setString(2,passwd);
                                rs=stmt.executeQuery();

                                if(rs.next()){
                                    outToClient.writeBytes("525 OK!\n");
```

```
                                System.out.println("服务端消息：525 OK!");
                                outToClient.flush();
                            }else{
                                outToClient.writeBytes(("401 ERROR!\n"));
                                System.out.println("服务端消息：401 ERROR!");
                                outToClient.flush();
                            }
                        }
                    }catch(Exception e){
                        System.out.println("statement 建立失败");
                    }
                    break;
                case "BAL":
                    operate_times++;
                    try{
                        if(conn!=null){
                            stmt=conn.prepareStatement("select balance from
info where ID=?");
                            stmt.setString(1,userID);
                            rs=stmt.executeQuery();

                            if(rs.next()) {
                                double bala = rs.getDouble("balance");
                                outToClient.writeBytes("AMNT:" + bala + "\n");
                                System.out.println("服务端消息：AMNT:" + bala);
                                outToClient.flush();
                            }else{
                                outToClient.writeBytes("401 ERROR!\n");
                                System.out.println("服务端消息：401 ERROR!");
                                outToClient.flush();
                            }
                        }
                    }catch(Exception e){
                        System.out.println("statement 建立失败");
                    }
                    break;
                case "WDR":
                    operate_times++;
                    String amountStr=substring(clientData,5);
                    try {
                        double amount = Double.parseDouble(amountStr);
                        try {
                            if (conn != null) {
                                stmt=conn.prepareStatement("select balance
from info where ID=?");
                                stmt.setString(1,userID);
                                rs = stmt.executeQuery();

                                rs.next();
                                double bala = rs.getDouble("balance");
                                double newBala = bala - amount;
                                if (newBala >= 0 && amount > 0 ) {//UPDATE 表名
称 SET 更新字段1=更新值1,更新字段2=更新值2,...[WHERE 更新条件(s)];
                                    stmt=conn.prepareStatement("update info set
balance=? where ID=?");
                                    stmt.setDouble(1,newBala);
                                    stmt.setString(2,userID);
                                    int ok_update = stmt.executeUpdate();
                                    //存储
                                    LocalDateTime dateTime =
```

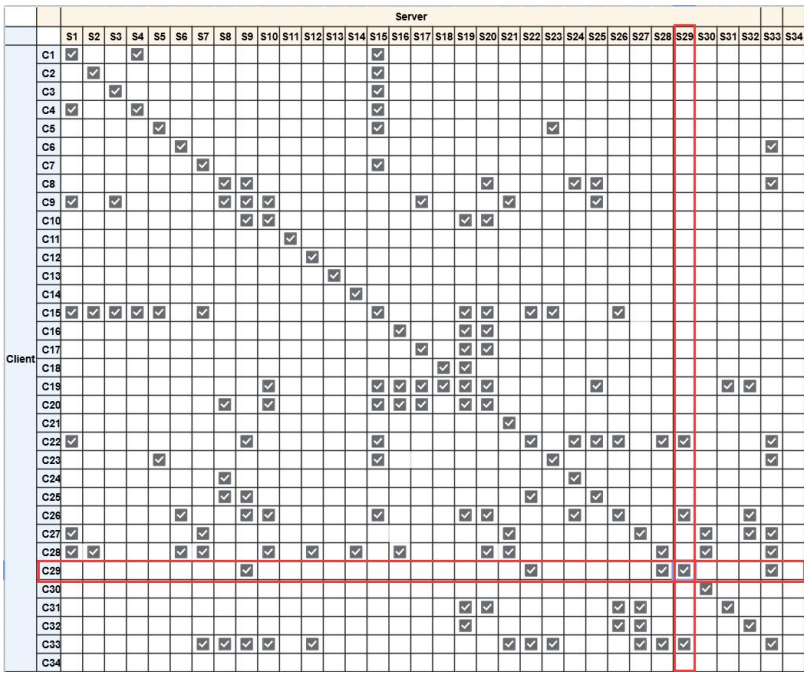
```
LocalDateTime.now();
                                    stmt=conn.prepareStatement("insert into
record_money(ID,money,datetime) values(?,?,'" + dateTime + "')");
                                    stmt.setString(1,userID);
                                    stmt.setDouble(2,amount);
                                    int ok_insert = stmt.executeUpdate();
                                    outToClient.writeBytes("525 OK!\n");
                                    System.out.println("服务端消息：525 OK!");
                                    outToClient.flush();
                                } else if(newBala<0){//余额不足
                                    outToClient.writeBytes("401 ERROR!\n");
                                    System.out.println("服务端消息：401 ERROR!");
                                    outToClient.flush();
                                }else{//非法输入，取出金额为负，你这是给我存钱呢？？？
                                    LocalDateTime dateTime =
LocalDateTime.now();
                                    stmt=conn.prepareStatement("insert into
error_logs values('" + dateTime + "',?,?)");
                                    stmt.setString(1,"error negative input");

stmt.setString(2,clientSocket.getInetAddress().getHostAddress());
                                    int ok_insert = stmt.executeUpdate();
                                    outToClient.writeBytes("401 ERROR!\n");
                                    System.out.println("服务端消息：401 ERROR!");
                                    outToClient.flush();
                                }

                            }
                        } catch (Exception e) {
                            System.out.println("statement 建立失败");
                        }
                        break;
                    }catch(Exception f){
                        try {
                            LocalDateTime dateTime = LocalDateTime.now();
                            if (conn != null) {
                                stmt = conn.prepareStatement("insert into
error_logs values('" + dateTime + "',?,?)");
                                stmt.setString(1, "error string input");
                                stmt.setString(2,
clientSocket.getInetAddress().getHostAddress());
                                int ok_insert = stmt.executeUpdate();
                                outToClient.writeBytes("401 ERROR!\n");
                                System.out.println("服务端消息：401 ERROR!");
                                outToClient.flush();
                                break;
                            }
                        }catch(Exception e){
                            System.out.println("statement 建立失败,错误日志无法存
储");
                        }
                    }
                case "BYE":
                    operate_times++;
                    outToClient.writeBytes("BYE\n");
                    System.out.println("服务端消息：BYE");
                    outToClient.flush();//清除输出流缓存，
                    // 关闭流和套接字
                    input.close();
                    outToClient.close();
```

```
                    clientSocket.close();
                    break;
                default:
                    outToClient.writeBytes("401 ERROR!\n");
                    System.out.println("服务端消息：401 ERROR!");
                    outToClient.flush();
            }
        }
    } catch (Exception e) {
        LocalDateTime dateTime = LocalDateTime.now();
        System.out.println("客户端连接已断开  "+dateTime+"\n\n\n");
    }
  }
}
```

四：交叉测试



六、结果及分析

在本次实验中，我们小组完成了 ATM 系统用户端与服务器端的初步搭建，同时也成功完成了组间交叉测试。在本次实验中，对于用户端成功完成了 GUI 界面，服务申请，输入错误验证等功能，能够保证系统的安全性，完整性同时也能成功向服务器发送请求，在交叉测试中也能顺利向其他小组服务器发送报文。而对于服务器方面，在完成基本客户端响应等功能的同时，我们还完成了用户错误输入检测以保证用户输入非法字符，并加入了日志记录等记录数据库操作，在交叉测试中也成功与其他小组连接并提供服务。

七、上机实验总结

在本次实验中，通过本次实验，我们小组认识到：

1：利用 socket 编程可以实现通过网络进行两个主机之间的交流，基于此可以完成许多系统设计的基础。

2：团队协作中每一个成员都需要积极配合小组成员，积极交流，在完成项目是要保证系统的完整性一致性，小组内应有共同的指导。

3：设计系统时，需主要考虑其安全性，可靠性，实用性，用户体验等。