

计算机网络课程作业二

ATM 模拟编程作业报告

1. 目的

通过作业二使同学们掌握所学 TCP 套接字编程方法，理解 TCP 套接字在服务器和客户端之间的应用层通信的流程和形式，巩固计算机网络的理论和知识。此外，通过作业的开展使同学们提高计算机网络程序设计和开发能力，熟悉所用编程语言的开发环境及调试过程，熟悉所用语言中的数据类型、数据结构、语法结构、运算方法等。基于作业一的内容进行程序设计开发，同学们也可以体会计算机网络通信协议(应用层)的制定、标准化等工作对实际应用的价值和意义。

2. 作业内容

要求两位同学组队(参考石墨文档 1-作业 2 分组)，基于模拟的应用层通信协议标准(参考石墨文档 2-“RFC20212021”)完成 ATM 机客户端和服务端端的设计和开发。程序语言不限。

客户端具备 GUI 可以模拟 ATM 机的账户登录和存款取款等功能，同时应该包括错误信息的提示。服务器端应该需要通过后台数据库维护账户信息、具备记录日志的功能。

各组同学独立完成，不能抄袭。作业结束时，每组需提交作业报告，其中包含程序设计思路和软件源代码。

里程碑 1：第 2 次实验课，完成组内程序设计开发和测试，尝试连接其他小组的服务器端。

里程碑 2：第 3 次实验课，分析解决里程碑 1 中的问题，完成程序调试，通过测试脚本完成对其他所有小组服务器端的连接，记录结果。

里程碑 3：第 4 次实验课，继续调试程序，最终提交文档、代码。

3. 运行环境

//在本节介绍使用的系统环境、编码环境、程序语言等内容，包括 TCP Socket 相关知识。

操作系统: Windows 11;

程序语言:

Java: Java 是一种高级编程语言, 它由 Sun Microsystems (后被 Oracle 收购) 于 1995 年推出。Java 的设计初衷是为了能够跨平台运行, 即开发一次, 到处运行。因此, 它的应用范围非常广泛, 包括桌面应用程序、Web 应用程序、移动应用程序、嵌入式系统等。Java 的语法类似于 C++, 但是去掉了 C++ 中的指针和操作符重载等复杂的概念, 并添加了垃圾回收机制, 使得 Java 更加安全和易于学习。Java 还提供了强大的 API (Application Programming Interface) 和各种类库, 使得开发者可以更加方便地实现各种功能。Java 应用程序通常会被编译成字节码文件, 这些文件可以在任何装有 Java 虚拟机 (JVM) 的机器上运行。这意味着 Java 应用程序可以轻松地在不同的操作系统和硬件平台上运行, 从而实现了 Java 的 “一次编写, 到处运行” 的特性。总之, Java 是一种强大的、灵活的编程语言, 它已经成为了计算机编程领域的主流之一, 广泛应用于各种领域。

系统环境:

Java 系统环境包括以下几个主要组成部分:

①Java Development Kit (JDK): 这是开发和编译 Java 应用程序所需的核心组件。它包含了 JRE (Java Runtime Environment)、编译器、调试器和其他开发工具。

②Java Runtime Environment (JRE): 这是运行 Java 应用程序所需的基本组件。它包含 Java 虚拟机 (JVM) 和 Java 类库等必要的组件。

③Java 虚拟机 (JVM): 这是 Java 应用程序的运行时引擎, 可以在不同的操作系统上实现 Java 应用程序的跨平台性。

④Java 类库: 这是一组预先编写的可重用代码模块, 包括各种 API (Application Programming Interface), 如 Swing、AWT、JDBC 等, 以帮助 Java 开发人员快速构建自己的应用程序。

⑤除此之外, 还有一些其他的 Java 环境相关工具和技术, 比如 Maven、Gradle、JUnit 等, 这些都可以帮助 Java 开发人员更好地开发和管理 Java 应用程序。

编码环境: IDE (Integrated Development Environment): IntelliJ IDEA。

相关理论:

TCP Socket 是一种基于 TCP/IP 协议的网络通信机制, 用于实现不同计算机之间的数据交换。其基本原理包括以下几个方面:

①IP 地址和端口号: 每台计算机都有唯一的 IP 地址, 用于标识不同的计算机。而端口号则用于区分一个计算机上不同的进程或应用程序。

②三次握手建立连接: 在进行 TCP Socket 通信之前, 需要通过三次握手协议来建立连接。这个过程包括客户端向服务器发送请求, 服务器接受请求并回复确认, 最后客户端再回复确认。

③数据传输: TCP Socket 会将要传输的数据进行分段, 并且对每个分段进行编号和校验, 以保证数据的可靠性和完整性。同时, 还会对数据进行流量控制和拥塞控制, 确保网络质量和带宽合理利用。

④四次挥手断开连接: 当数据传输完成后, 需要通过四次挥手协议来断开连接。这个过程包括客户端向服务器发送请求断开连接, 服务器回复确认, 服务器向客户端发送断开连接请求, 客户端回复确认, 最终完成断开连接的操作。

TCP Socket 的工作方式如下：

①一个程序通过创建 Socket 对象来打开一个端口，并等待来自另一台计算机的连接请求。每个连接都有一个唯一的 Socket 对象。当客户端连接到服务器时，它将打开一个 Socket，并分配一个本地端口号，该端口将用于与服务器进行通信。

②一旦 Socket 已经打开，客户端和服务器之间就可以通过读取从对方发送的数据流来传递数据。这些数据可以是任何类型的，从简单的文本信息到二进制文件或图像。发送数据时，应用程序将数据写入 Socket 的输出流中，而接收数据时则从 Socket 的输入流中读取数据。

TCP Socket 具有以下优点：

①可靠性：TCP Socket 使用基于确认和重传的机制来确保数据的可靠传输。如果发送的数据包未能到达目标，则会立即进行重发。

②顺序性：TCP Socket 确保数据包按照其发送顺序进行接收，从而避免了数据包乱序的问题。

③全双工通信：TCP Socket 允许客户端和服务器在同一时间进行双向通信，从而使得通信更加高效。

④流控制：TCP Socket 可以通过动态调整发送数据的速率来避免网络拥塞和资源浪费。

4. 程序设计

4.1 设计思路

//在本节介绍程序的流程、客户端和服务端端的交互流程图等

RFC-20212021

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

2. Messages from Server to ATM

Msg name	Purpose
500 sp AUTH REQUIRE	Ask user for PIN (password)
525 sp OK!	Requested operation (PASSWD, WITHDRAWL) OK
401 sp ERROR!	requested operation (PASSWD, WITHDRAWL) in ERROR
AMNT:<amnt>	Sent in response to BALANCE request
BYE	User done, display welcome screen at ATM

3. Interaction between ATM and Server:

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

4. Server port number

2525

这段 Java 程序的设计思路是实现一个基于客户端-服务器架构的 ATM 取款系统。具体来说，该系统由客户端和服务端两部分组成，客户端负责与用户进行交互，而服务器则处理客户端发送的请求并返回响应。

在设计时，需要考虑以下几个要素：

①消息传输：客户端向服务器发送请求，服务器对请求进行处理后返回相应的响应消息。根据规定的交互协议，客户端和服务端之间需要通过网络传输消息，实现双向通信；

②交互协议：客户端和服务端之间的交互遵循一定的协议，包括 HELO、PASS、BALA、WDRA 和 BYE 等命令。客户端向服务器发送请求时需要按照规定格式发送指令，服务器接收到指令后需要解析处理，并返回相应的响应消息；

③用户身份验证：客户端需要先验证用户的身份信息（如 userid 和 passwd），以确保只有合法用户能够使用该系统。同时，为了加强安全性，密码需要进行加密存储；

④数据库操作：服务器需要连接数据库，从中获取用户的账户余额等信息，以便通过 BALA 命令查询余额，或者在 WDRA 命令中判断是否有足够的余额支持取款操作。如果取款操作成功，则需要更新数据库中的账户余额信息；

⑤系统可靠性：为了确保系统的稳定性和可靠性，客户端和服务端之间需要建立长连接，并设定心跳机制以防止连接超时。同时，在出现异常情况时也需要进行相应的错误处理，防止系统崩溃或数据丢失。

总体来说，该程序的设计思路是基于传统的客户端-服务器架构，采用 TCP/IP 协议进行网络通信，使用规范化的交互协议，通过连接数据库实现数据的持久化存储。同时，在实现过程中需要考虑安全性、可靠性等方面的问题，以确保系统能够满足用户需求并具有一定的可扩展性。

4.2 服务器端

//服务器端源代码，应加以文字进行简要说明

4.2.1 Mysql. java（连接数据库）

```
1 package Work;
2
3 import java.sql.*;
4
5 public class Mysql {
6     //连接信息
7     private static String driverName = "com.mysql.cj.jdbc.Driver";
8     private static String url = "jdbc:mysql://localhost:3306/net_ex?useUnicode=true&characterEncoding=utf8&useSSL=true";
9     private static String username = "root"; // 数据库用户名
10    private static String password = "123456"; // 数据库密码
11
12    //jdbc对象
13    private Connection connection = null; // 连接对象
14    private PreparedStatement preparedStatement = null; // SQL语句预处理对象
15    private ResultSet resultSet = null; // 结果集对象
16
17    //获取连接
18    public void getConnection() {
19        try {
20            //2、建立连接
21            Class.forName(driverName); // 加载MySQL数据库的JDBC驱动程序
22            connection = DriverManager.getConnection(url, username, password); // 创建与MySQL数据库的连接
23        } catch (SQLException e) {
24            e.printStackTrace();
25        } catch (ClassNotFoundException e) {
26            e.printStackTrace();
27        }
28    }
29
30    //取钱操作
31    public void takeaway(int take, String UserId) {
32        try {
33            getConnection(); // 获取数据库连接
34            String sql = "UPDATE atm set Balance = Balance - ? Where UserId = ?"; // 更新余额的SQL语句
35            preparedStatement = connection.prepareStatement(sql);
36            preparedStatement.setInt(1, take); // 设置第一个占位符的值为取款金额
37            preparedStatement.setString(2, UserId); // 设置第二个占位符的值为用户ID
38            preparedStatement.executeUpdate(); // 执行SQL更新操作
39        } catch (SQLException e) {
40            e.printStackTrace();
41        }
42    }
43 }
```

```

44 //查询操作
45 public ResultSet select(String sql) {
46     try {
47         getConnection(); // 获取数据库连接
48         preparedStatement = connection.prepareStatement(sql); // 创建预处理对象，使用传入的SQL语句
49         resultSet = preparedStatement.executeQuery(); // 执行查询操作，并返回结果集
50     } catch (SQLException e) {
51         e.printStackTrace();
52     }
53     return resultSet; // 返回结果集
54 }
55
56 //检查用户ID是否存在
57 public boolean checkUserId(String UserId) {
58     Mysql db = new Mysql();
59     boolean check = false;
60     resultSet = db.select( sql: "Select * from atm where UserId = " + UserId); // 查询用户ID是否存在的SQL语句
61     try {
62         while (resultSet.next()) {
63             String Id = resultSet.getString( columnLabel: "UserId");
64             if (UserId.equals(Id)){
65                 check = true;
66             }
67         }
68     } catch (SQLException e) {
69         e.printStackTrace();
70     }
71     return check; // 返回检查结果
72 }
73

```

```

74 //检查密码是否正确
75 public boolean checkPass(String UserId,String Password) {
76     Mysql db = new Mysql();
77     boolean check = false;
78     resultSet = db.select( sql: "Select * from atm where UserId = " + UserId); // 查询用户密码是否正确的SQL语句
79     try {
80         while (resultSet.next()) {
81             String pass = resultSet.getString( columnLabel: "PassWord");
82             if (Password.equals(pass)){
83                 check = true;
84             }
85         }
86     } catch (SQLException e) {
87         e.printStackTrace();
88     }
89     return check; // 返回检查结果
90 }
91
92 //查询余额
93 public int sltBalance(String UserId){
94     Mysql db = new Mysql();
95     int balance = 0;
96     resultSet = db.select( sql: "Select * from atm where UserId = " + UserId); // 查询用户余额的SQL语句
97     try {
98         while (resultSet.next()) {
99             balance = resultSet.getInt( columnLabel: "Balance"); // 获取查询结果中的余额数据
100         }
101     } catch (SQLException e) {
102         e.printStackTrace();
103     }
104     return balance; // 返回余额值
105 }
106

```

```

107 //断开连接
108 public void closeConnection() {
109     //5、断开连接
110     if (resultSet != null) {
111         try {
112             resultSet.close(); // 关闭ResultSet对象
113         } catch (SQLException e) {
114             e.printStackTrace();
115         }
116     }
117     if (preparedStatement != null) {
118         try {
119             preparedStatement.close(); // 关闭PreparedStatement对象
120         } catch (SQLException e) {
121             e.printStackTrace();
122         }
123     }
124     if (connection != null) {
125         try {
126             connection.close(); // 关闭Connection对象
127         } catch (SQLException e) {
128             e.printStackTrace();
129         }
130     }
131 }
132 }

```

这段代码实现的是一个 Java 程序连接 MySQL 数据库的类，主要包含以下方法：

- ①getConnection()方法，用于获取与 MySQL 数据库的连接；
- ②takeaway(int take, String UserId)方法，用于将用户从 ATM 机取出一定金额；
- ③select(String sql)方法，用于执行 SQL 查询语句并返回结果集；
- ④checkUserId(String UserId)方法，用于检查给定的用户 ID 是否存在；
- ⑤checkPass(String UserId,String Password)方法，用于检查给定的用户密码是否正确；
- ⑥sltBalance(String UserId)方法，用于查询给定用户的余额；
- ⑦closeConnection()方法，用于关闭所有打开的连接、预处理对象和结果集对象。

总体来说，这个类封装了与数据库的交互细节，使得业务逻辑代码可以更加简洁明了。同时，使用 PreparedStatement 可以有效防止 SQL 注入攻击。

不过，该类中的一些方法可能存在一定的安全漏洞，例如 checkUserId()方法和 checkPass()方法中的 SQL 查询语句没有使用参数化查询，容易受到 SQL 注入攻击，应该在后续的开发过程中加以改进。

4.2.2 TCPServer.java (服务器端源码)

```
1 package Work;
2
3 import java.io.*;
4 import java.net.*;
5 import java.text.SimpleDateFormat;
6 import java.util.Date;
7 import java.util.Scanner;
8
9 public class TCPServer {
10     public static void main(String[] args) throws Exception {
11         // 创建一个ServerSocket，监听来自客户端的连接
12         String UserId = ""; // 用户ID
13         String passWord; // 用户密码
14         int cunkuan = 0; // 存储用户余额
15         MySQL db = new MySQL(); // 创建MySQL数据库对象
16         ServerSocket serverSocket = new ServerSocket(2525); // 创建服务器套接字并指定其监听端口
17         System.out.println("服务器已启动，等待客户端连接...");
18         Socket socket = serverSocket.accept(); // 接受客户端的连接请求，并返回表示新连接的Socket对象
19         System.out.println("客户端连接成功!");
20         FileWriter write = new FileWriter("C:\\Users\\13299\\Desktop\\Log.txt", append: true); // 创建FileWriter对象，用于向日志文件写入数据
21         BufferedWriter bw = new BufferedWriter(write); // 创建BufferedWriter对象，用于缓冲输出流并提高效率
22         InetAddress address = socket.getInetAddress(); // 获取客户端的IP地址
23         SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"); // 创建日期格式化对象
24         Date date;
25
26         try {
27             bw.write("连接成功，客户端的IP: " + address.getHostAddress() + "\n"); // 将客户端IP写入日志文件中
28         } catch (IOException e) {
29             e.printStackTrace();
30         }
31
32         while (true) {
33             String cmd; // 存储客户端输入的命令
34             // 读取来自客户端的数据
35             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream())); // 创建BufferedReader对象，用于从Socket中读取数据
36             PrintWriter out = new PrintWriter(socket.getOutputStream(), true); // 创建PrintWriter对象，用于向Socket中写入数据
37             String str = in.readLine(); // 读取一行客户端输入，并存储到字符串变量str中
38             Scanner reader = new Scanner(str); // 创建Scanner对象，用于解析客户端输入的命令
39             cmd = reader.next(); // 获取客户端输入的第一个参数
40
```

```
41         date = new Date(System.currentTimeMillis());
42         if (cmd.equals("HELLO")) {
43             UserId = reader.next(); // 获取用户ID
44             boolean checkUserId = db.checkUserId(UserId); // 检查用户ID是否存在
45             try {
46                 bw.write(sdf.format(date) + " Input: HELLO " + UserId + "\n"); // 将用户输入的命令和参数写入日志文件中
47             } catch (IOException e) {
48                 e.printStackTrace();
49             }
50             if (checkUserId) {
51                 out.println("500 AUTH REQUIRED!"); // 如果用户ID已存在，则返回错误信息
52                 try {
53                     bw.write(sdf.format(date) + " Output: 500 AUTH REQUIRED!\n"); // 将服务器返回的结果写入日志文件中
54                 } catch (IOException e) {
55                     e.printStackTrace();
56                 }
57             } else {
58                 out.println("402 USER DOESN'T EXIST"); // 如果用户ID不存在，则返回错误信息
59                 try {
60                     bw.write(sdf.format(date) + " Output: 401 ERROR!\n"); // 将服务器返回的结果写入日志文件中
61                 } catch (IOException e) {
62                     e.printStackTrace();
63                 }
64             }
65         }
66     }
67 }
```

```

67     date = new Date(System.currentTimeMillis());
68     if (cmd.equals("PASS")){
69         passWord = reader.next(); // 获取用户密码
70         boolean checkPass = db.checkPass(UserId,passWord); // 检查用户密码是否正确
71         try {
72             bw.write( str: sdf.format(date) + " Input: PASS " + passWord + "\n"); // 将用户输入的命令和参数写入日志文件
73         }catch(IOException e){
74             e.printStackTrace();
75         }
76         if (checkPass){
77             out.println("525 OK!"); // 如果密码正确，则返回成功信息
78             try {
79                 bw.write( str: sdf.format(date) + " Output: 525 OK!\n"); // 将服务器返回的结果写入日志文件中
80             }catch(IOException e){
81                 e.printStackTrace();
82             }
83         }else {
84             out.println("401 ERROR!"); // 如果密码错误，则返回错误信息
85             try {
86                 bw.write( str: sdf.format(date) + " Output: 401 ERROR!\n"); // 将服务器返回的结果写入日志文件中
87             }catch(IOException e){
88                 e.printStackTrace();
89             }
90         }
91     }
92
93     date = new Date(System.currentTimeMillis());
94     if (cmd.equals("BALA")){
95         cunkuan = db.sltBalance(UserId); // 查询用户余额
96         try {
97             bw.write( str: sdf.format(date) + " Input: BALA\n"); // 将用户输入的命令和参数写入日志文件中
98         }catch(IOException e){
99             e.printStackTrace();
100         }
101         out.println("AMNT:" + cunkuan ); // 返回用户余额
102         try {
103             bw.write( str: sdf.format(date) + " Output: AMNT:" + cunkuan + "\n"); // 将服务器返回的结果写入日志文件中
104         }catch(IOException e){
105             e.printStackTrace();
106         }
107     }

```

```

109     date = new Date(System.currentTimeMillis());
110     if (cmd.equals("WDRA")){
111         int takeway = Integer.parseInt(reader.next()); // 获取用户从ATM机取款金额
112         try {
113             bw.write( str: sdf.format(date) + " Input: WDRA " + takeway + "\n"); // 将用户输入的命令和参数写入日志文件
114         }catch(IOException e){
115             e.printStackTrace();
116         }
117         if (takeway <= cunkuan){
118             out.println("525 OK!"); // 如果用户余额充足，则返回成功信息，并更新用户账户余额
119             try {
120                 bw.write( str: sdf.format(date) + " Output: 525 OK!\n"); // 将服务器返回的结果写入日志文件中
121             }catch(IOException e){
122                 e.printStackTrace();
123             }
124             db.takeaway(takeway,UserId);
125         }else {
126             out.println("401 ERROR!"); // 如果用户余额不足，则返回错误信息
127             try {
128                 bw.write( str: sdf.format(date) + " Output: 401 ERROR!\n"); // 将服务器返回的结果写入日志文件中
129             }catch(IOException e){
130                 e.printStackTrace();
131             }
132         }
133     }

```

```

134
135     date = new Date(System.currentTimeMillis());
136     if (cmd.equals("BYE")){
137         try {
138             bw.write( str: sdf.format(date) + "   Input: BYE\n"); // 将用户输入的命令和参数写入日志文件中
139         }catch(IOException e){
140             e.printStackTrace();
141         }
142         System.out.println("感谢使用!");
143         out.println("BYE"); // 返回退出信息
144         try {
145             bw.write( str: sdf.format(date) + "   Output: BYE\n"); // 将服务器返回的结果写入日志文件中
146         }catch(IOException e){
147             e.printStackTrace();
148         }
149         break;
150     }
151 }
152 db.closeConnection(); // 关闭数据库连接
153 System.out.println("数据库连接已断开!");
154 socket.close(); // 关闭客户端连接
155 System.out.println("客户端连接已断开!");
156 try {
157     bw.write( str: "连接已断开\n"); // 将连接关闭信息写入日志文件中
158 }catch(IOException e){
159     e.printStackTrace();
160 }
161 bw.close(); // 关闭BufferedWriter对象
162 write.close(); // 关闭FileWriter对象
163 System.out.println("日志已断开!");
164 }
165 }

```

该代码是一个简单的 TCP 服务器程序，能够接受客户端的连接并处理来自客户端的命令。该程序主要实现了如下功能：

- ①导入必要的包；
- ②定义主类 TCPServer，并在 main 函数中创建 ServerSocket 对象，监听来自客户端的连接；
- ③定义一些变量和对象，如用户 ID、密码、余额等，以及日期格式化对象、MySQL 数据库对象、日志文件输出流等；
- ④在程序中不断循环，等待客户端发送数据，并对客户端发送的命令进行处理；
- ⑤对不同的命令进行处理，包括 HELO、PASS、BALA、WDRA 和 BYE；
- ⑥在处理命令时，将相关信息写入日志文件中；
- ⑦在程序结束时，关闭一些资源，如数据库连接、客户端连接和日志输出流等；
- ⑧对不同的命令进行处理时，向客户端发送不同的响应信息；
- ⑨在程序结束时关闭资源，包括数据库连接、客户端连接和日志输出流等；
- ⑩程序中使用了 MySQL 数据库，通过 Mysql 类来实现对数据库的操作。

4.3 客户端

//客户端源代码，应加以文字进行简要说明

4.3.1 clientSwing.java (客户端 UI 界面)

```
1 package Work;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class clientSwing {
7     JFrame jf = new JFrame( title: "ATM"); // 创建名为"ATM"的JFrame对象
8     int set = -1; // 初始化变量set为-1
9     String text1; // 存储文本框中的内容
10    String text2;
11    String text3;
12    String text4;
13    JLabel label01 = new JLabel(); // 创建用于显示标签的JLabel对象
14    JLabel label02 = new JLabel();
15    JLabel label03 = new JLabel();
16    JLabel label04 = new JLabel();
17    JLabel label05 = new JLabel();
18    JLabel label06 = new JLabel();
19    JLabel label07 = new JLabel();
20    final JTextField textField1 = new JTextField( columns: 8); // 创建用于输入文本的JTextField对象
21    final JPasswordField textField2 = new JPasswordField( columns: 8);
22    final JTextField textField3 = new JTextField( columns: 8);
23    final JTextField textField4 = new JTextField( columns: 8);
24    JButton btn1 = new JButton( text: "提交"); // 创建用于点击的按钮
25    JButton btn2 = new JButton( text: "提交");
26    JButton btn3 = new JButton( text: "提交");
27    JButton btn4 = new JButton( text: "退出");
28    JButton btn5 = new JButton( text: "提交");
29    JPanel panel = new JPanel( layout: null); // 创建使用空布局的JPanel对象
30
31    void build(){
32        jf.setSize( width: 500, height: 300); // 设置窗口大小
33        jf.setLocationRelativeTo(null); // 将窗口定位到屏幕中央
34        jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); // 点击关闭按钮时关闭程序
35
36        label01.setText("ATM取款系统"); // 设置标签的文本内容和字体
37        label01.setFont(new Font( name: null, Font.PLAIN, size: 25)); // 设置字体, null 表示使用默认字体
38        label02.setText("UserID:");
39        label02.setFont(new Font( name: null, Font.PLAIN, size: 18));
40        label03.setText("PassWord:");
41        label03.setFont(new Font( name: null, Font.PLAIN, size: 18));
42        label04.setText("取款:");
43        label04.setFont(new Font( name: null, Font.PLAIN, size: 18));
44        label05.setText("账户:");
45        label05.setFont(new Font( name: null, Font.PLAIN, size: 18));
46        label06.setText("存款:");
47        label06.setFont(new Font( name: null, Font.PLAIN, size: 18));
48        label07.setText("IP:");
49        label07.setFont(new Font( name: null, Font.PLAIN, size: 18));
50
51        label01.setBounds( x: 170, y: 0, width: 200, height: 25); // 设置标签的位置和大小
52        label02.setBounds( x: 10, y: 60, width: 100, height: 25);
53        label03.setBounds( x: 10, y: 90, width: 100, height: 25);
54        label04.setBounds( x: 10, y: 120, width: 100, height: 25);
55        label05.setBounds( x: 10, y: 150, width: 300, height: 25);
56        label06.setBounds( x: 10, y: 180, width: 300, height: 25);
57        label07.setBounds( x: 10, y: 30, width: 300, height: 25);
58
59        panel.add(label01); // 将标签添加到panel中
60        panel.add(label02);
61        panel.add(label03);
62        panel.add(label04);
63        panel.add(label05);
64        panel.add(label06);
65        panel.add(label07);
```

```

68 // 设置文本框，指定可见列数为8列
69 textField1.setFont(new Font( name: null, Font.PLAIN, size: 20));
70 textField2.setFont(new Font( name: null, Font.PLAIN, size: 20));
71 textField3.setFont(new Font( name: null, Font.PLAIN, size: 20));
72 textField4.setFont(new Font( name: null, Font.PLAIN, size: 20));
73 textField4.setText("127.0.0.1"); // 设置IP地址文本框的默认值
74
75 textField1.setBounds( x: 120, y: 60, width: 200, height: 25); // 设置文本框的位置和大小
76 textField2.setBounds( x: 120, y: 90, width: 200, height: 25);
77 textField3.setBounds( x: 120, y: 120, width: 200, height: 25);
78 textField4.setBounds( x: 120, y: 30, width: 200, height: 25);
79
80 panel.add(textField1); // 将文本框添加到panel中
81 panel.add(textField2);
82 panel.add(textField3);
83 panel.add(textField4);
84
85
86 // 设置按钮，点击后获取文本框中的文本
87 btn1.setFont(new Font( name: null, Font.PLAIN, size: 20)); // 设置字体
88 btn2.setFont(new Font( name: null, Font.PLAIN, size: 20));
89 btn3.setFont(new Font( name: null, Font.PLAIN, size: 20));
90 btn4.setFont(new Font( name: null, Font.PLAIN, size: 20));
91
92 btn5.setFont(new Font( name: null, Font.PLAIN, size: 20));
93
94
95 btn1.setBounds( x: 330, y: 60, width: 100, height: 25); //x坐标10，y坐标0，组件宽100，高50
96 btn2.setBounds( x: 330, y: 90, width: 100, height: 25);
97 btn3.setBounds( x: 330, y: 120, width: 100, height: 25);
98 btn4.setBounds( x: 330, y: 150, width: 100, height: 25);
99 btn5.setBounds( x: 330, y: 30, width: 100, height: 25);
100
101 panel.add(btn1); // 将按钮添加到panel中
102 panel.add(btn2);
103 panel.add(btn3);
104 panel.add(btn4);
105 panel.add(btn5);
106
107
108 jf.setContentPane(panel); // 将panel设置为窗口的内容面板
109 jf.setVisible(true); // 显示窗口
110 }
111
112 public static void main(String[] args) {
113     clientSwing swing = new clientSwing(); // 创建clientSwing对象
114     swing.build(); // 调用build方法创建界面
115 }
116 }

```

这段 Java 代码主要是创建了一个 ATM 取款系统的图形用户界面，可以方便地进行 ATM 取款操作。该界面使用了 Swing 库，包括 JFrame、JPanel、JLabel、JButton 和 JPasswordField 等组件。在代码中，首先创建了一个名为 "ATM" 的 JFrame 对象，并初始化了一些变量和文本框控件。随后在 build() 方法中，将各种控件添加到 JPanel 中，并设置它们的位置和大小等属性。最后将该 JPanel 设为 JFrame 的内容面板，并显示窗口。

具体来说，该程序实现了以下功能：

- ①设置了标签和文本框控件，用于输入账户信息、密码、取款金额、账户号码、存款金额和 IP 地址等；
- ②设置了按钮控件，分别用于提交账户信息、密码、取款金额、账户号码、存款金额和退出程序；
- ③设置了窗口的标题、大小和位置，以及关闭窗口时退出程序。

4.3.2 TCPClient.java (客户端源码)

```
1 package Work;
2
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5 import java.io.*;
6 import java.net.*;
7 import java.util.Scanner;
8
9 public class TCPClient {
10     public static void main(String[] args) throws Exception {
11         //建立窗口
12         final clientSwing swing = new clientSwing();
13         swing.build();
14         String IP;
15
16         //设置按钮监听器
17         swing.btn5.addActionListener(new ActionListener() {
18             public void actionPerformed(ActionEvent e) {
19                 swing.text4 = swing.textField4.getText();
20                 swing.set = 0;
21             }
22         });
23
24         //等待用户输入IP地址
25         while (swing.set == -1){
26             Thread.sleep(1000);
27         }
28
29         //获取用户输入的IP地址
30         IP = swing.text4;
31
32         //创建Socket对象，连接服务器
33         final Socket socket = new Socket(IP, 2525);
34         System.out.println("与服务器连接成功!");
35         swing.label07.setText("IPV");
36
37         //设置按钮监听器，实现登录功能
38         swing.btn1.addActionListener(new ActionListener() {
39             public void actionPerformed(ActionEvent e) {
40                 if (swing.set == 0) {
41                     swing.text1 = swing.textField1.getText();
42                     String str = "HELLO " + swing.text1;
43                     PrintWriter out = null;
44                     try {
45                         out = new PrintWriter(socket.getOutputStream(), true);
46                     } catch (IOException ioException) {
47                         ioException.printStackTrace();
48                     }
49                     out.println(str);
50                     BufferedReader inFromServer = null;
51                     try {
52                         inFromServer = new BufferedReader(new InputStreamReader(socket.getInputStream()));
53                     } catch (IOException ioException) {
54                         ioException.printStackTrace();
55                     }
56                     String back = null;
57                     try {
58                         back = inFromServer.readLine();
59                     } catch (IOException ioException) {
60                         ioException.printStackTrace();
61                     }
62                     Scanner reader = new Scanner(back);
63                     if (reader.next().equals("500")) {
64                         swing.label05.setText("账户: " + swing.text1);
65                         swing.set = 1;
66                         swing.label02.setText("UserIdv");
67                     } else {
68                         swing.label05.setText("账户: 不存在该用户");
69                         swing.set = 0;
70                     }
71                 }
72             }
73         });
74     }
75 }
```

```

75 //设置按钮监听器，实现密码验证功能以及查询余额
76 swing.btn2.addActionListener(new ActionListener() {
77     public void actionPerformed(ActionEvent e) {
78         swing.text2 = swing.textField2.getText();
79         String str = "PASS " + swing.text2;
80         if (swing.set == 1 || swing.set == 2){
81             PrintWriter out = null;
82             try {
83                 out = new PrintWriter(socket.getOutputStream(), autoFlush: true);
84             } catch (IOException ioException) {
85                 ioException.printStackTrace();
86             }
87             out.println(str);
88             BufferedReader inFromServer = null;
89             try {
90                 inFromServer = new BufferedReader(new InputStreamReader(socket.getInputStream()));
91             } catch (IOException ioException) {
92                 ioException.printStackTrace();
93             }
94             String back = null;
95             try {
96                 back = inFromServer.readLine();
97             } catch (IOException ioException) {
98                 ioException.printStackTrace();
99             }
100             Scanner reader = new Scanner(back);
101
102             if (reader.next().equals("525")) {
103                 swing.label05.setText("账户: " + swing.text1 + " 登录成功");
104                 swing.set = 2;
105                 swing.label03.setText("PassWordv");
106                 str = "BALA";
107                 out.println(str);
108                 try {
109                     back = inFromServer.readLine();
110                 } catch (IOException ioException) {
111                     ioException.printStackTrace();
112                 }
113                 reader = new Scanner(back);
114                 String amt = reader.next();
115                 if (amt.startsWith("AMNT:")) {
116                     swing.label06.setText("存款: ¥" + amt.substring(beginIndex: 5));
117                 } else {
118                     swing.label06.setText("存款: 发生错误!");
119                 }
120             } else {
121                 swing.label06.setText("密码: 密码错误!");
122             }
123         } else if (swing.set == 0){
124             swing.label05.setText("账户: 请先输入UserId!");
125         }
126     }
127 });

```

```

128 //设置按钮监听器，实现取款功能
129 swing.btn3.addActionListener(new ActionListener() {
130     public void actionPerformed(ActionEvent e) {
131         swing.text3 = swing.textField3.getText();
132         String str = "WDRA " + swing.text3;
133         if (swing.set == 2){
134             PrintWriter out = null;
135             try {
136                 out = new PrintWriter(socket.getOutputStream(), autoFlush: true);
137             } catch (IOException ioException) {
138                 ioException.printStackTrace();
139             }
140             out.println(str);
141             BufferedReader inFromServer = null;
142             try {
143                 inFromServer = new BufferedReader(new InputStreamReader(socket.getInputStream()));
144             } catch (IOException ioException) {
145                 ioException.printStackTrace();
146             }
147             String back = null;

```

```

148         try {
149             back = inFromServer.readLine();
150         } catch (IOException ioException) {
151             ioException.printStackTrace();
152         }
153         Scanner reader = new Scanner(back);
154         if (reader.next().equals("525")) {
155             str = "BALA";
156             out.println(str);
157             try {
158                 back = inFromServer.readLine();
159             } catch (IOException ioException) {
160                 ioException.printStackTrace();
161             }
162             reader = new Scanner(back);
163             String amt = reader.next();
164             if (amt.startsWith("AMNT:")) {
165                 swing.label06.setText("存款: ¥" + amt.substring(beginIndex: 5));
166             } else {
167                 swing.label06.setText("存款: 发生错误!");
168             }
169         } else {
170             swing.label06.setText("存款: 金额不足!");
171         }
172     } else {
173         swing.label06.setText("存款: 请先登录!");
174     }
175 }
176 });

```

```

177
178 //设置按钮监听器, 实现退出功能
179 swing.btn4.addActionListener(new ActionListener() {
180     public void actionPerformed(ActionEvent e) {
181         String str = "BYE";
182         PrintWriter out = null;
183         try {
184             out = new PrintWriter(socket.getOutputStream(), autoFlush: true);
185         } catch (IOException ioException) {
186             ioException.printStackTrace();
187         }
188         out.println(str);
189         BufferedReader inFromServer = null;
190         try {
191             inFromServer = new BufferedReader(new InputStreamReader(socket.getInputStream()));
192         } catch (IOException ioException) {
193             ioException.printStackTrace();
194         }
195         String back = null;
196         try {
197             back = inFromServer.readLine();
198         } catch (IOException ioException) {
199             ioException.printStackTrace();
200         }
201         Scanner reader = new Scanner(back);
202         if (reader.next().equals("BYE")) {
203             swing.jf.dispose(); //关闭窗口
204             try {
205                 socket.close(); //关闭Socket连接
206             } catch (IOException ioException) {
207                 ioException.printStackTrace();
208             }
209             System.out.println("与服务器的连接已断开!");
210         }
211     }
212 });
213 }
214 }

```

这是一个 TCP 客户端的 Java 代码。在主函数中，程序首先创建了一个图形用户界面（GUI），然后设置若干个按钮监听器实现不同的功能。具体来说，程序通过 Socket 对象连接服务器并发送不同的命令、接收服务器返回的信息，实现了以下四个功能：

①登录功能：用户需要输入 IP 地址和 UserId，程序将其发送给服务器进行验证。如果验证通过，程序会显示账户名称，并将状态码设为 1；

②密码验证与查询余额功能：用户需要输入密码，程序将其发送给服务器进行验证。如果验证通过，程序会查询账户余额并将其显示出来；

③取款功能：用户需要输入取款金额，程序将其发送给服务器进行处理。如果余额足够，则服务器会扣除相应金额并将新余额返回给客户端；

④退出功能：用户可以点击按钮退出程序，关闭 GUI 并断开与服务器的连接。

5. 测试运行结果

//测试运行过程中的记录，包括截图和文字描述、问题分析、改进方案等。


详情请见以下内容~

5.1 组内测试

//在客户端使用手工录入的方式，测试验证所有的程序功能。

5.1.1 登录

5.1.1.1 登录界面



ATM取款系统

IP:	<input type="text" value="127.0.0.1"/>	<input type="button" value="提交"/>
UserID:	<input type="text"/>	<input type="button" value="提交"/>
PassWord:	<input type="text"/>	<input type="button" value="提交"/>
取款:	<input type="text"/>	<input type="button" value="提交"/>
账户:		<input type="button" value="退出"/>
存款:		

默认自动填入的 IP 地址为“127.0.0.1”（本机地址）；

5.1.1.2 输入服务器端 IP 地址(IPv4)并成功连接



ATM取款系统

IP ✓ 10.234.113.34 提交

UserID: 提交

PassWord: 提交

取款: 提交

账户: 退出

存款:

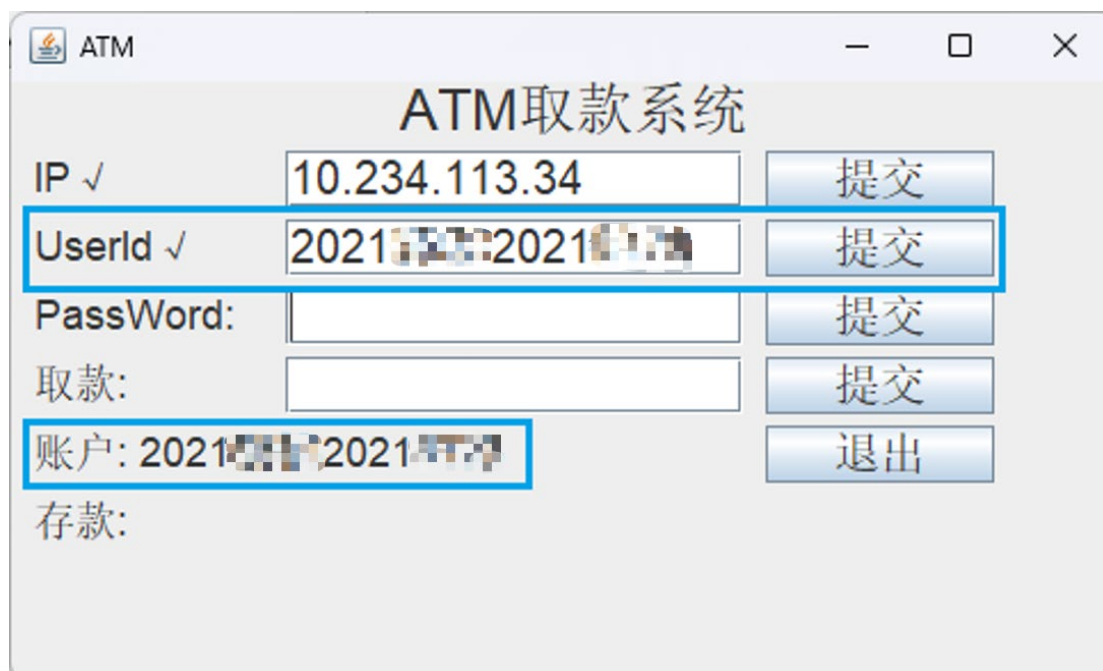
经查询得本次服务器端 IP 地址为：10.234.113.34，输入后点击对应的“提交”按钮，左侧的“IP”标签会变成“IP ✓”，表示连接成功，此时服务器端有如下反馈：

服务器已启动，等待客户端连接...
客户端连接成功！

客户端有如下反馈：

与服务器连接成功！

5.1.1.3 输入正确的用户账号



ATM取款系统

IP ✓ 10.234.113.34 提交

Userld ✓ 2021 提交

PassWord: 提交

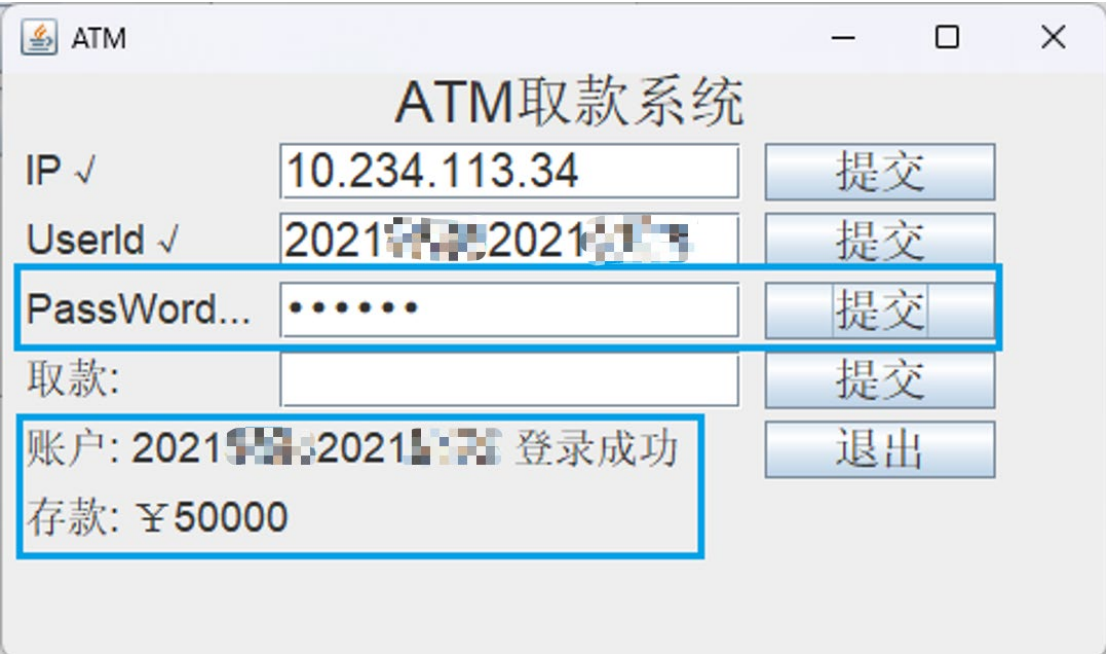
取款: 提交

账户: 2021 退出

存款:

输入后点击对应的“提交”按钮，左侧的“UserId”标签会变成“UserId ✓”，表示用户账号正确，下面出现账户号码；

5.1.1.4 输入匹配的密码

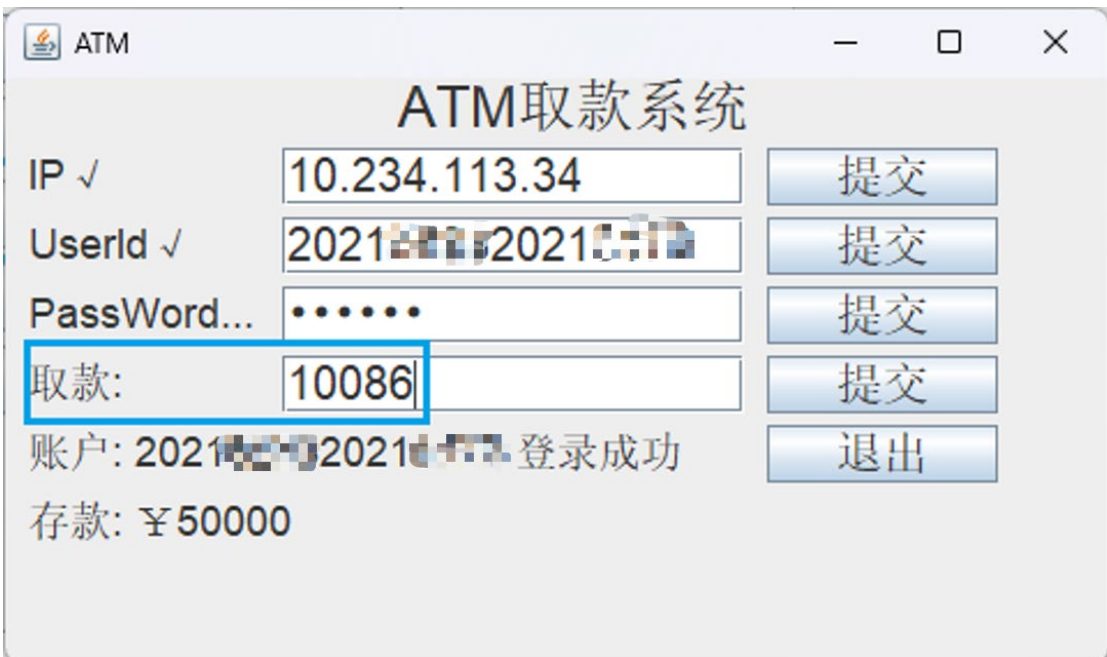


The screenshot shows a window titled "ATM" with the subtitle "ATM取款系统". It contains several input fields and buttons. The "IP" field is filled with "10.234.113.34" and has a checkmark next to its label. The "UserId" field is filled with "2021" followed by masked characters and has a checkmark next to its label. The "PassWord..." field is filled with masked characters and is highlighted with a blue border. To the right of each input field is a "提交" (Submit) button. Below the password field is a "取款:" (Withdrawal) field and another "提交" button. At the bottom, there is a status area showing "账户: 2021" followed by masked characters and "登录成功" (Login successful), and "存款: ¥50000" (Balance: ¥50000). A "退出" (Exit) button is also present.

输入后点击对应的“提交”按钮，左侧的“PassWord”标签会变成“Password ✓”，表示用户账号和密码匹配正确，下面出现账户“登陆成功”的字样，并显示存款余额；

5.1.2 取款

5.1.2.1 输入取款金额



This screenshot is similar to the previous one, but the "取款:" (Withdrawal) field is now filled with "10086" and is highlighted with a blue border. The "PassWord..." field is no longer highlighted. The status area at the bottom still shows "账户: 2021" followed by masked characters and "登录成功" (Login successful), and "存款: ¥50000" (Balance: ¥50000). The "退出" (Exit) button is still present.

输入后点击对应的“提交”按钮；

5.1.2.2 成功取款

ATM取款系统

IP ✓ 10.234.113.34 提交

UserId ✓ 2021... 提交

PassWord... 提交

取款: 10086 提交

账户: 2021... 登录成功 退出

存款: ¥39914

下面的存款余额正确更新 ($50000 - 10086 = 39914$);

查看服务器端数据库:

①取款前:

	UserId	PassWord	Balance
	2021...2021...	123456	50000
	2021...2021...	123456	50000
	2021...2021...	123456	50000
	2021...2021...	123456	50000
	2021...2021...	123456	50000
	2021...2021...	123456	50000
	2021...2021...	123456	50000
	2021...2021...	123456	50000
	2021...2021...	123456	50000
	2021...DZ24...	123456	50000

②取款后:

	UserId	PassWord	Balance
	2021-01-01 2021-01-01	123456	50000
	2021-01-01 2021-01-01	123456	50000
	2021-01-01 2021-01-01	123456	50000
	2021-01-01 2021-01-01	123456	50000
	2021-01-01 2021-01-01	123456	50000
	2021-01-01 2021-01-01	123456	50000
	2021-01-01 2021-01-01	123456	50000
	2021-01-01 2021-01-01	123456	50000
	2021-01-01 2021-01-01	123456	39914
	2021-01-01 2021-01-01	123456	50000

5.1.3 退出

ATM

ATM取款系统

IP ✓

10.234.113.34

提交

UserId ✓

2021-01-01 2021-01-01

提交

PassWord...

.....

提交

取款:

10086

提交

账户: 2021-01-01 2021-01-01 登录成功

退出

存款: ¥39914

在任意阶段点击“退出”按钮即可退出该系统，关闭界面；
服务器端有如下反馈：

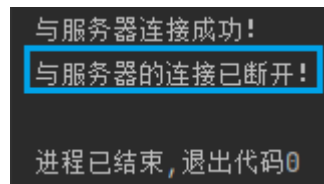
```

服务器已启动，等待客户端连接...
客户端连接成功！
感谢使用！
数据库连接已断开！
客户端连接已断开！
日志已断开！

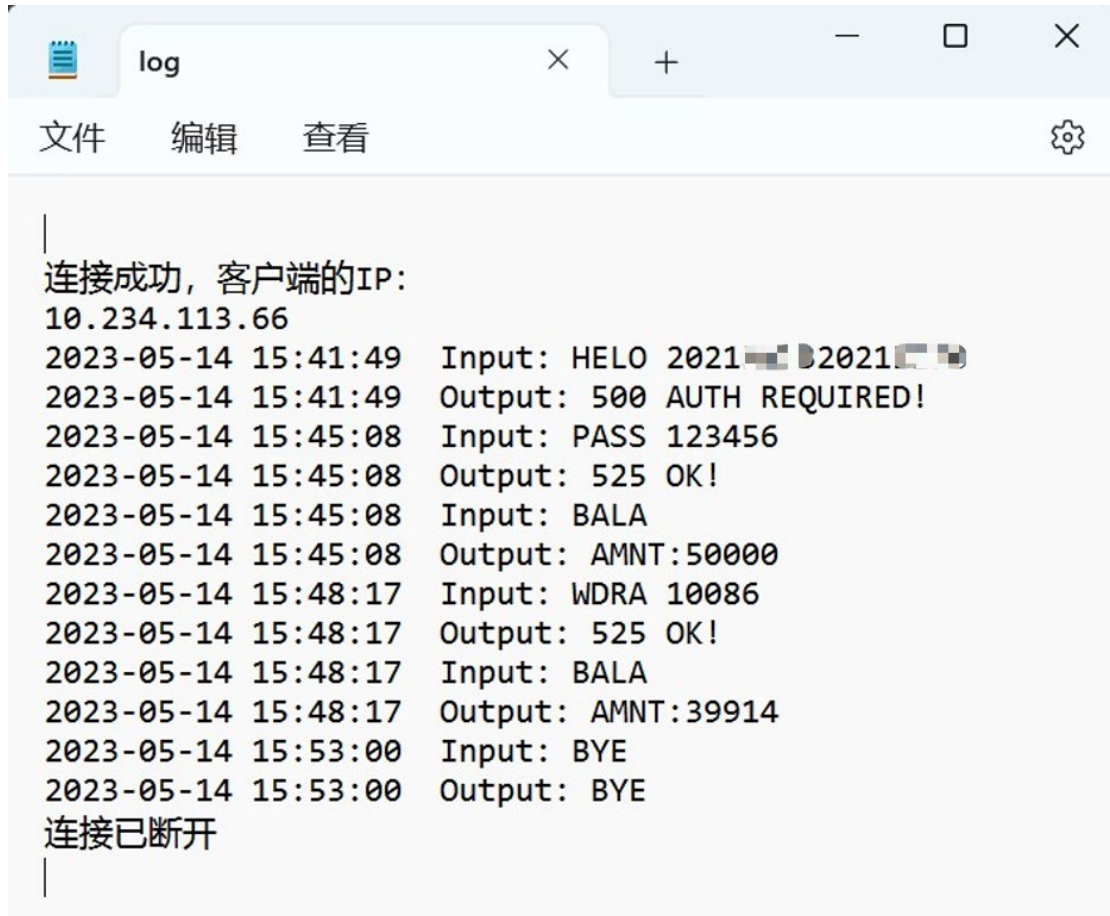
Process finished with exit code 0

```


客户端有如下反馈：

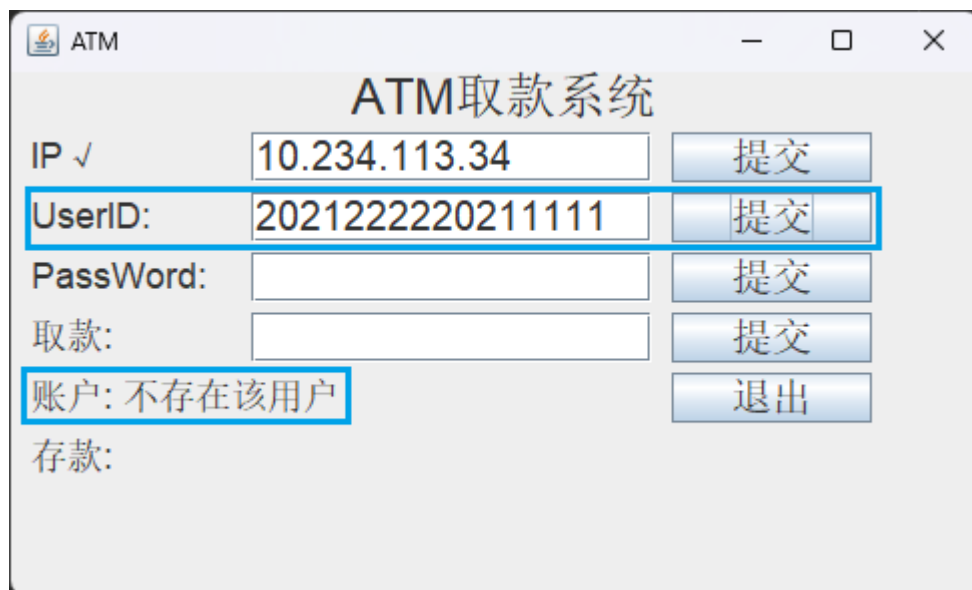


5.1.4 日志文件



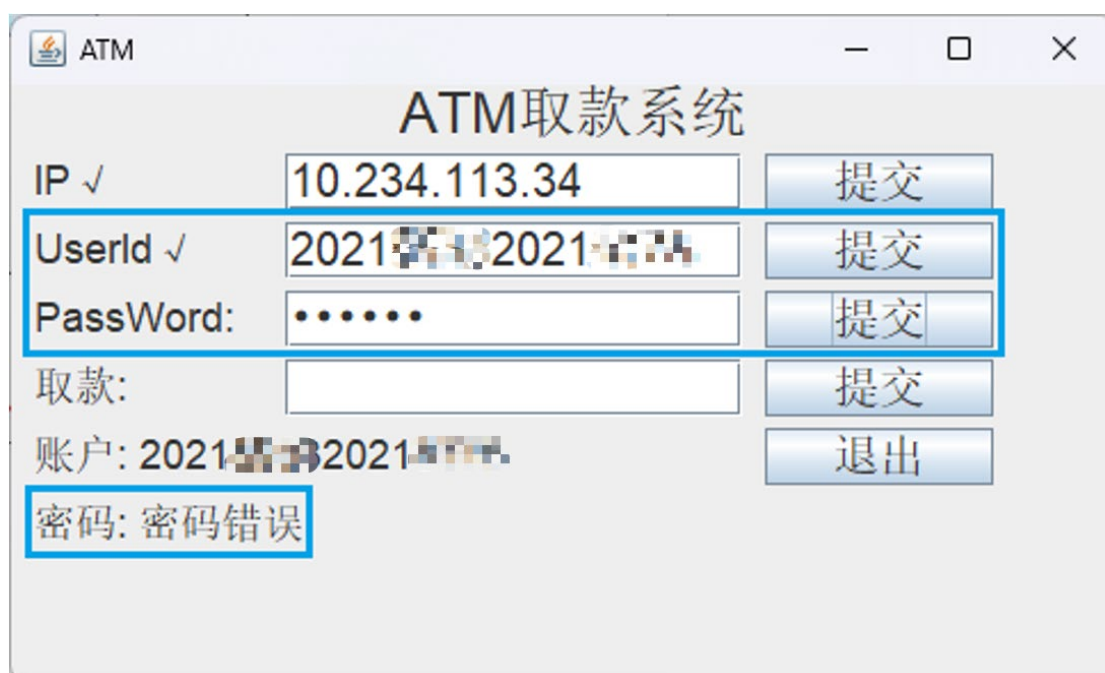
5.1.5 异常处理

5.1.5.1 输入用户账号错误



输入错误的用户账号后点击对应的“提交”按钮，左侧的“UserID”标签并未变成“UserID ✓”，且下面出现账户“不存在该用户”的字样，表示输入账号错误；

5.1.5.2 输入密码错误



输入正确的用户账号和错误的密码后，点击对应的“提交”按钮，左侧的“Password”标签并未变成“Password ✓”，表示用户账号和密码匹配失败，下面出现密码“密码错误”的字样，表示输入密码错误；

5.1.5.3 取款金额大于账户余额



ATM取款系统

IP ✓ 10.234.113.34 提交

UserId ✓ 2021 2021 提交

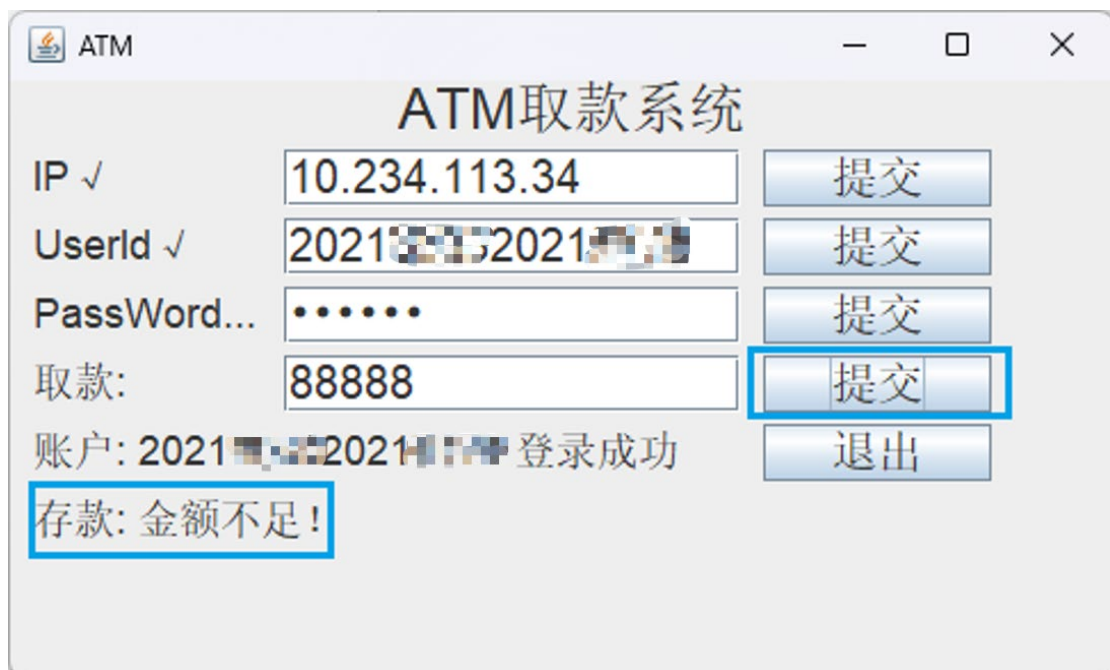
PassWord... 提交

取款: 88888 提交

账户: 2021 2021 登录成功 退出

存款: ¥39914

成功登录后，输入大于账户余额的取款金额；



ATM取款系统

IP ✓ 10.234.113.34 提交

UserId ✓ 2021 2021 提交

PassWord... 提交

取款: 88888 提交

账户: 2021 2021 登录成功 退出

存款: 金额不足!

点击对应的“提交”后按钮，下面出现存款“金额不足！”的字样，表示取款金额大于账户余额，取款失败。

5.1.5.4 日志文件

连接成功，客户端的IP:

10.234.113.66

```
2023-05-14 16:03:37 Input: HELO 2021222220211111
2023-05-14 16:03:37 Output: 401 ERROR!
2023-05-14 16:06:04 Input: HELO 2021-2021-
2023-05-14 16:06:04 Output: 500 AUTH REQUIRED!
2023-05-14 16:06:08 Input: PASS 111111
2023-05-14 16:06:08 Output: 401 ERROR!
2023-05-14 16:08:52 Input: PASS 123456
2023-05-14 16:08:52 Output: 525 OK!
2023-05-14 16:08:52 Input: BALA
2023-05-14 16:08:52 Output: AMNT:39914
2023-05-14 16:08:58 Input: WDRA 88888
2023-05-14 16:08:58 Output: 401 ERROR!
2023-05-14 16:09:03 Input: PASS 123456
2023-05-14 16:09:03 Output: 525 OK!
2023-05-14 16:09:03 Input: BALA
2023-05-14 16:09:03 Output: AMNT:39914
2023-05-14 16:10:34 Input: WDRA 88888
2023-05-14 16:10:34 Output: 401 ERROR!
2023-05-14 16:13:35 Input: BYE
2023-05-14 16:13:35 Output: BYE
```

连接已断开

5.2 组间测试

//在组内测试通过的基础上，可以在本组的客户端程序中写入自动化的测试脚本，连接其他小组的服务器进行测试。在测试结果的基础上逐渐改进客户端和服务器端程序。

5.2.1 测试对象：

9			2021-2021-	123456	¥50,000.00
---	--	--	------------	--------	------------

5.2.2 我方作为 Client，对方作为 Server

5.2.2.1 连接服务器



The screenshot shows the initial state of the 'ATM取款系统' (ATM Withdrawal System) window. The title bar includes an icon and the text 'ATM'. The window title is 'ATM取款系统'. The interface contains several input fields and buttons:

- IP:** A label followed by a checkmark and a text box containing '10.234.113.38'. A blue box highlights this row.
- UserID:** A label followed by an empty text box and a '提交' (Submit) button.
- PassWord:** A label followed by an empty text box and a '提交' (Submit) button.
- 取款:** A label followed by an empty text box and a '提交' (Submit) button.
- 账户:** A label followed by an empty text box and a '退出' (Exit) button.
- 存款:** A label followed by an empty text box.

经查询得对方服务器端 IP 地址为：10.234.113.38，输入后点击对应的“提交”按钮，左侧的“IP”标签会变成“IP ✓”，表示连接成功，客户端有如下反馈：

与服务器连接成功！

5.2.2.2 成功登录



The screenshot shows the 'ATM取款系统' window after a successful login. The title bar and window title remain the same. The interface now displays the following information:

- IP:** The label is now 'IP ✓' and the text box contains '10.234.113.38'.
- Userld:** The label is now 'Userld ✓' and the text box contains '2021' followed by a masked area.
- PassWord...** The label is followed by a text box containing six dots '.....'.
- 取款:** The label is followed by an empty text box.
- 账户:** The label is followed by '2021' followed by a masked area, and the text '登录成功' (Login Successful) is displayed.
- 存款:** The label is followed by '¥ 50000'.

The buttons '提交' (Submit) and '退出' (Exit) are still present next to the respective fields.

5.2.2.3 取款

ATM

— □ ×

ATM取款系统

IP ✓

10.234.113.38

提交

UserId ✓

2021 2021

提交

PassWord...

.....

提交

取款:

11111

提交

账户: 2021 2021 登录成功

退出

存款: ¥38889

	UserId	PassWord	Balance
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
▶	2021 2021	123456	50000
	2021 2021	123456	50000
	UserId	PassWord	Balance
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	50000
	2021 2021	123456	38889
	2021 2021	123456	50000

5.2.3 我方作为 Server，对方作为 Client

5.2.3.1 成功登录

bankLogin

ATM取款系统

UserID: 2021 2021 提交

PassWord: 提交

欢迎您, 账户: 2021 2021

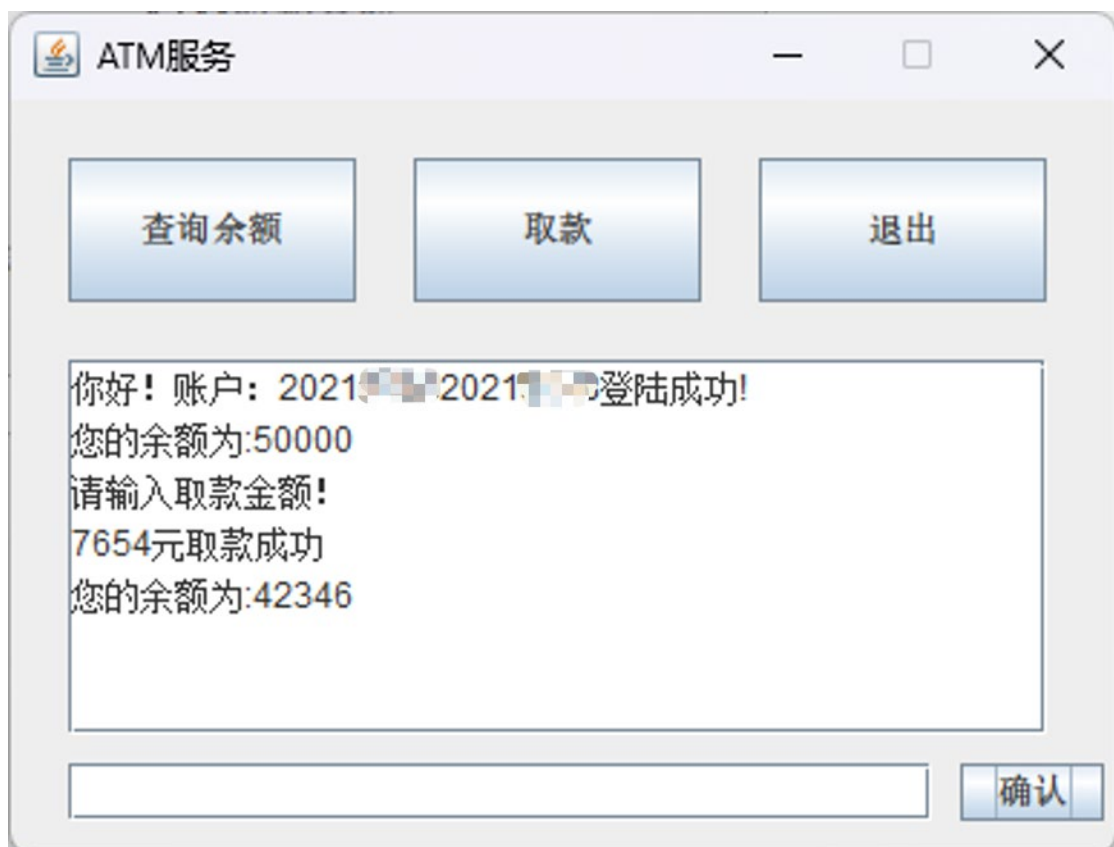
ATM服务

查询余额 取款 退出

你好! 账户: 2021 2021 登陆成功!
您的余额为:50000

确认

5.2.3.2 取款



2021...2021...	123456	50000
2021...2021...	123456	50000
2021...DZ220...	123456	50000

2021...2021...	123456	50000
2021...2021...	123456	42346
2021...DZ220...	123456	50000

5.2.3.3 客户端反馈

```

服务器连接成功!
HELO 2021...2021...

AMNT:42346
服务器连接已断开!

进程已结束,退出代码0
  
```


5.2.3.4 服务器端反馈 & 日志文件

```
服务器已启动，等待客户端连接...
客户端连接成功！
感谢使用！
数据库连接已断开！
客户端连接已断开！
日志已断开！

Process finished with exit code 0
```

连接成功，客户端的IP:

10.234.113.50

2023-05-14 17:05:04 Input: HELO 2021 2021

```
2023-05-14 17:05:04  Output: 500 AUTH REQUIRED!
```

2023-05-14 17:05:15 Input: PASS 123456

2023-05-14 17:05:15 Output: 525 OK!

2023-05-14 17:05:15 Input: BALA

2023-05-14 17:05:15 Output: AMNT:50000

2023-05-14 17:05:48 Input: WDRA 7654

2023-05-14 17:05:48 Output: 525 OK!

2023-05-14 17:05:48 Input: BALA

2023-05-14 17:05:48 Output: AMNT:42346

2023-05-14 17:06:16 Input: BALA

2023-05-14 17:06:16 Output: AMNT:42346

2023-05-14 17:06:25 Input: BYE

2023-05-14 17:06:25 Output: BYE

连接已断开

6. 总结

//介绍一下开展作业 1、2 相关工作遇到的问题、解决的思路，以及相关的收获等。

在开展这项作业相关工作时，我们遇到了以下几个问题：

①开发语言选择：各位同学擅长的编程语言不同，且各种编程语言解决问题的方式有所不同，我们经过讨论和比较后最终选择了 Java 作为我们的开发语言。

②数据库的设计：服务器端需要通过后台数据库维护账户信息，我们需要对数据库进行设计。由于组内成员对数据库的经验不足，我们花费了大量时间进行学习和研究，最终设计出了符合需求的数据库。

③网络连接问题：在尝试连接其他小组的服务器端时，我们遇到了一些网络连接问题。我们利用 Wireshark 等工具进行抓包分析，最终定位并解决了问题。

针对以上问题，我们采取了以下解决思路：

①开发语言选择：我们考虑到 Java 是一门功能强大的程序语言，并且拥有丰富的类库，可以快速实现我们的需求。同时，我们也结合了本学期开设的另外一门课程——Java EE 程序设计，利用课上学习的关于 Swing 类和连接本地数据库相应的知识，让组内成员能够快速上手；

②数据库的设计：我们先收集了 ATM 机业务的相关数据，然后根据数据特点和需求，设计出了符合要求的数据库结构，并使用 MySQL Workbench 8.0 CE 进行实现，我们还编写了相应的 SQL 语句对数据库进行操作；

③网络连接问题：我们首先检查了代码中的错误，并使用调试工具进行调试。当发现无法解决时，我们利用 Wireshark 等网络抓包工具对网络数据进行分析，找出问题所在并尝试修复。

在完成这项作业相关工作后，我们获得了以下收获：

①编程能力的提升：通过这次作业，我们学习了网络编程、图形用户界面设计、数据库设计等知识，提升了我们的编程能力和实践经验；

②团队协作能力的提高：在团队中，我们需要互相协作、交流和合作，这次作业让我们更好地锻炼了团队协作能力；

③自我管理能力的提升：由于这项作业需要每位成员都有一定的自主开发能力，我们需要掌握时间管理、任务分配等方面的技巧，提升自我管理能力；

④加深计算机网络知识的理解：通过本次作业我们掌握了所学 TCP Socket 编程方法，理解了 TCP Socket 在服务器和客户端之间的应用层通信的流程和形式，巩固了计算机网络的理论和知识。