**成都信息工程学院计算机学院**

**课**

**程**

**实**

**验**

**报**

**告**

|  |  |
| --- | --- |
| **实验课程：** | **网络游戏编程技术** |
| **实验项目：** | **注册与登录** |
| **指导教师：** |  |
| **学生姓名:** |  |
| **学生学号：** |  |
| **班 级：** |  |
| **实验地点：** |  |
| **实验时间：** | **20 24 年 11 月 20 日 16：00 点~ 17:40点** |
| **实验成绩：** |  |

## 一【上机实验内容】

1. 为单机游戏增加场景，能连接课程群中的服务器，并保持在线。具体应用层协议见实验指导书；

2. 完成客户端的注册功能；

3. 完成客户端的登录功能；

4. 以上2、3两条，控制用户输入用户名时，不能出现空格。

## 二【上机实验步骤】（重点）

### 操作步骤

1.编辑客户端代码，使客户端能成功连接服务器。

2.编辑客户端代码，使客户端能在注册后成功返回对应字符串。

3.编辑客户端代码，使客户端在登录后能成功返回对应字符串。

4.编辑服务端代码，使服务端能正确接收客户端的字符串并做出对应反应。

### 关键代码

服务端代码：

1. package com.cy.socket;  
     
   import com.cy.mapper.UserMapper;  
   import com.cy.pojo.User;  
   import com.cy.util.MD5Util;  
   import com.cy.util.MybatisUtil;  
   import org.apache.ibatis.session.SqlSession;  
     
   import java.io.BufferedReader;  
   import java.io.IOException;  
   import java.io.InputStreamReader;  
   import java.io.PrintWriter;  
   import java.net.ServerSocket;  
   import java.net.Socket;  
   import java.util.HashSet;  
   import java.util.Scanner;  
   import java.util.Set;  
     
   public class ChatServer {  
    private static final int *PORT* = 10002;  
    private static final Set<ClientHandler> *clientHandlers* = new HashSet<>();  
    private static SqlSession *sqlSession* = MybatisUtil.*sqlSessionFactory*();  
    private static UserMapper *userMapper* = *sqlSession*.getMapper(UserMapper.class);  
     
    public static void main(String[] args) {  
    *startServer*();  
    *listenForMessages*();  
    }  
     
    public static void startServer() {  
    System.*out*.println("服务器启动，监听端口：" + *PORT*);  
    new Thread(() -> {  
    try (ServerSocket serverSocket = new ServerSocket(*PORT*)) {  
    while (true) {  
    Socket clientSocket = serverSocket.accept();  
    ClientHandler handler = new ClientHandler(clientSocket);  
    synchronized (*clientHandlers*) {  
    *clientHandlers*.add(handler);  
    System.*out*.println("客户端已添加，当前数量为" + *clientHandlers*.size());  
    }  
    handler.start();  
    }  
    } catch (IOException e) {  
    e.printStackTrace();  
    }  
    }).start();  
    }  
     
    public static void listenForMessages() {  
    new Thread(() -> {  
    try (Scanner scanner = new Scanner(System.*in*)) {  
    while (true) {  
    String message = scanner.nextLine();  
    *sendMessageToAllClients*(message);  
    }  
    }  
    }).start();  
    }  
     
    public static void sendMessageToAllClients(String message) {  
    synchronized (*clientHandlers*) {  
    System.*out*.println("当前连接的客户端数量: " + *clientHandlers*.size());  
    for (ClientHandler handler : *clientHandlers*) {  
    handler.sendMessage(message);  
    // 打印客户端信息  
    System.*out*.println("消息发送到客户端: " + handler.getId());  
    }  
    }  
    }  
     
    private static class ClientHandler extends Thread {  
    private Socket socket;  
    private PrintWriter out;  
    private BufferedReader in;  
     
    public ClientHandler(Socket socket) {  
    this.socket = socket;  
    System.*out*.println("新客户端连接: " + socket.getInetAddress().getHostAddress() + ":" + socket.getPort());  
    try {  
    in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
    out = new PrintWriter(socket.getOutputStream(), true);  
    } catch (IOException e) {  
    e.printStackTrace();  
    }  
    }  
     
    public void run() {  
    try {  
    String message;  
    while ((message = in.readLine()) != null) {  
    System.*out*.println("收到消息: " + message);  
    String s1 = message.split(" ")[0];  
    //注册操作  
    if ("Regist".equals(s1.replace("\uFEFF", ""))) {  
    String password = MD5Util.*getMD5*(message.split(" ")[2].replace("\uFEFF", ""));  
    String username = message.split(" ")[1].replace("\uFEFF", "");  
    String ip = socket.getInetAddress().getHostAddress() + ":" + socket.getPort();  
    User user = *userMapper*.select(username);  
    //用户名存在  
    if (user != null) {  
    sendMessage("RegistFail 1 ");  
    System.*out*.println("用户名存在");  
    } else {  
    User user1 = new User();  
    user1.setUsername(username);  
    user1.setPassword(password);  
    user1.setOnline(1);  
    user1.setScore(0);  
    user1.setUserIp(ip);  
    *userMapper*.regist(user1);  
    *sqlSession*.commit();  
    sendMessage("RegistSuccess ");  
    System.*out*.println("注册成功");  
    }  
    } else if (message.split(" ")[0].replace("\uFEFF", "").equals("Login")) {//登录操作  
    String password = MD5Util.*getMD5*(message.split(" ")[2].replace("\uFEFF", ""));  
    String username = message.split(" ")[1].replace("\uFEFF", "");  
    User user = *userMapper*.select(username);  
    //登录成功  
    if (user != null && user.getPassword().equals(password)) {  
    String ip = socket.getInetAddress().getHostAddress() + ":" + socket.getPort();  
    *userMapper*.online(username, ip);  
    *sqlSession*.commit();  
    sendMessage("LoginSuccess");  
    System.*out*.println("登录成功");  
    } else if (user == null) {  
    sendMessage("LoginFail 1");  
    System.*out*.println("用户不存在");  
    } else if (!user.getPassword().equals(MD5Util.*getMD5*(password))) {  
    sendMessage("LoginFail 2");  
    System.*out*.println("密码错误");  
    } else if (user.getOnline() == 1) {  
    sendMessage("LoginFail 3");  
    System.*out*.println("用户已登录");  
    }  
    }  
    //这会把消息重新发送给所有客户端  
   // broadcast(message);  
    }  
    } catch (IOException e) {  
    e.printStackTrace();  
    } finally {  
    try {  
    synchronized (*clientHandlers*) {  
    *clientHandlers*.remove(this);  
    String ip = socket.getInetAddress().getHostAddress() + ":" + socket.getPort();  
    User user = *userMapper*.findByIp(ip);  
    if (user != null) {  
    *userMapper*.updateByIp(ip);  
    *sqlSession*.commit();  
    }  
    System.*out*.println("客户端数量为" + *clientHandlers*.size());  
    }  
    socket.close();  
    } catch (IOException e) {  
    e.printStackTrace();  
    }  
     
    }  
    }  
     
    public void sendMessage(String message) {  
    out.println(message);  
    }  
     
    private void broadcast(String message) {  
    synchronized (*clientHandlers*) {  
    for (ClientHandler handler : *clientHandlers*) {  
    handler.sendMessage(message);  
    }  
    }  
    }  
    }  
   }

客户端代码：

using System;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using System.Net.Sockets;

using System.Text;

using TMPro;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class Netword\_u1 : MonoBehaviour

{

TMP\_InputField usernameInputField;

TMP\_InputField passwordInputField;

TMP\_Text outputText;

Socket clientSocket;

byte[] recvBuff = new byte[1024];

string recvStr;

byte[] sendBuff;

NetworkStream networkStream;

StreamWriter writer;

StreamReader reader;

bool isConnected = false;

// Start is called before the first frame update

void Start()

{

usernameInputField = GameObject.Find("Canvas/UsernameInput").GetComponent<TMP\_InputField>(); // 获取 InputField 组件

passwordInputField = GameObject.Find("Canvas/PasswordInput").GetComponent<TMP\_InputField>(); // 获取 InputField 组件

outputText = GameObject.Find("Canvas/OutputText").GetComponent<TMP\_Text>();

}

public void ButtonClicked()

{

clientSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

try

{

clientSocket.Connect("localhost", 10002);

networkStream = new NetworkStream(clientSocket);

writer = new StreamWriter(networkStream, Encoding.UTF8);

reader = new StreamReader(networkStream, Encoding.UTF8);

isConnected = true;

GameObject.Find("Canvas/ConnectBtn").GetComponent<Button>().enabled = false;

StartCoroutine(ReceiveData());

}

catch (System.Exception)

{

throw;

}

}

public void RegistBtnClicked()

{

if (isConnected && writer != null)

{

try

{

string username = usernameInputField.text;

string password = passwordInputField.text;

string sendStr = "Regist " + username + " " + password;

writer.WriteLine(sendStr);

writer.Flush();

Debug.Log("发送消息" + sendStr);

}

catch (Exception e)

{

Debug.Log("消息发送失败" + e.Message);

}

}

}

public void LoginBtnClicked()

{

if (isConnected && writer != null)

{

try

{

string username = usernameInputField.text;

string password = passwordInputField.text;

string sendStr = "Login " + username + " " + password;

writer.WriteLine(sendStr);

writer.Flush();

Debug.Log("发送消息" + sendStr);

}

catch (Exception e)

{

Debug.Log("消息发送失败" + e.Message);

}

}

}

private IEnumerator ReceiveData()

{

while (isConnected)

{

if (networkStream.DataAvailable)

{

try

{

recvStr = reader.ReadLine();

if (!string.IsNullOrEmpty(recvStr))

{

Debug.Log("收到服务器消息: " + recvStr);

string[] args = recvStr.Split(' ');

if (args[0] == "RegistSuccess")

recvStr = "regist success";

if (args[0] == "RegistFail")

recvStr = "username access";

if (args[0] == "LoginFail")

{

switch (args[1])

{

case "1":

recvStr = "username no access, please regist";

break;

case "2":

recvStr = "password error";

break;

case "3":

recvStr = "has login";

break;

}

}

if (args[0] == "LoginSuccess")

SceneManager.LoadScene(1);

outputText.text = recvStr;

}

}

catch (Exception e)

{

Debug.LogError("接收消息失败: " + e.Message);

isConnected = false;

}

}

yield return null;

}

}

// Update is called once per frame

void Update()

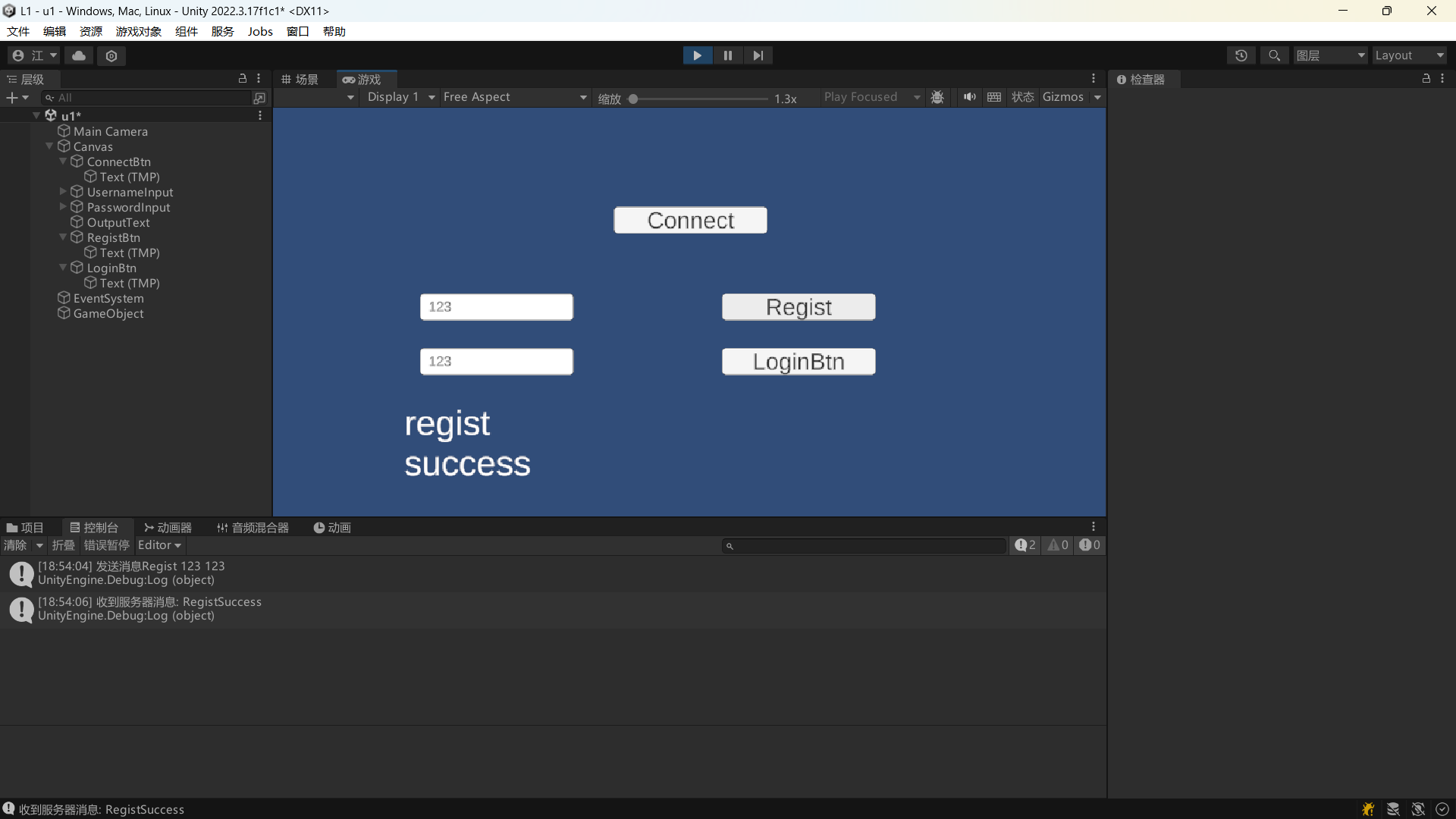
{

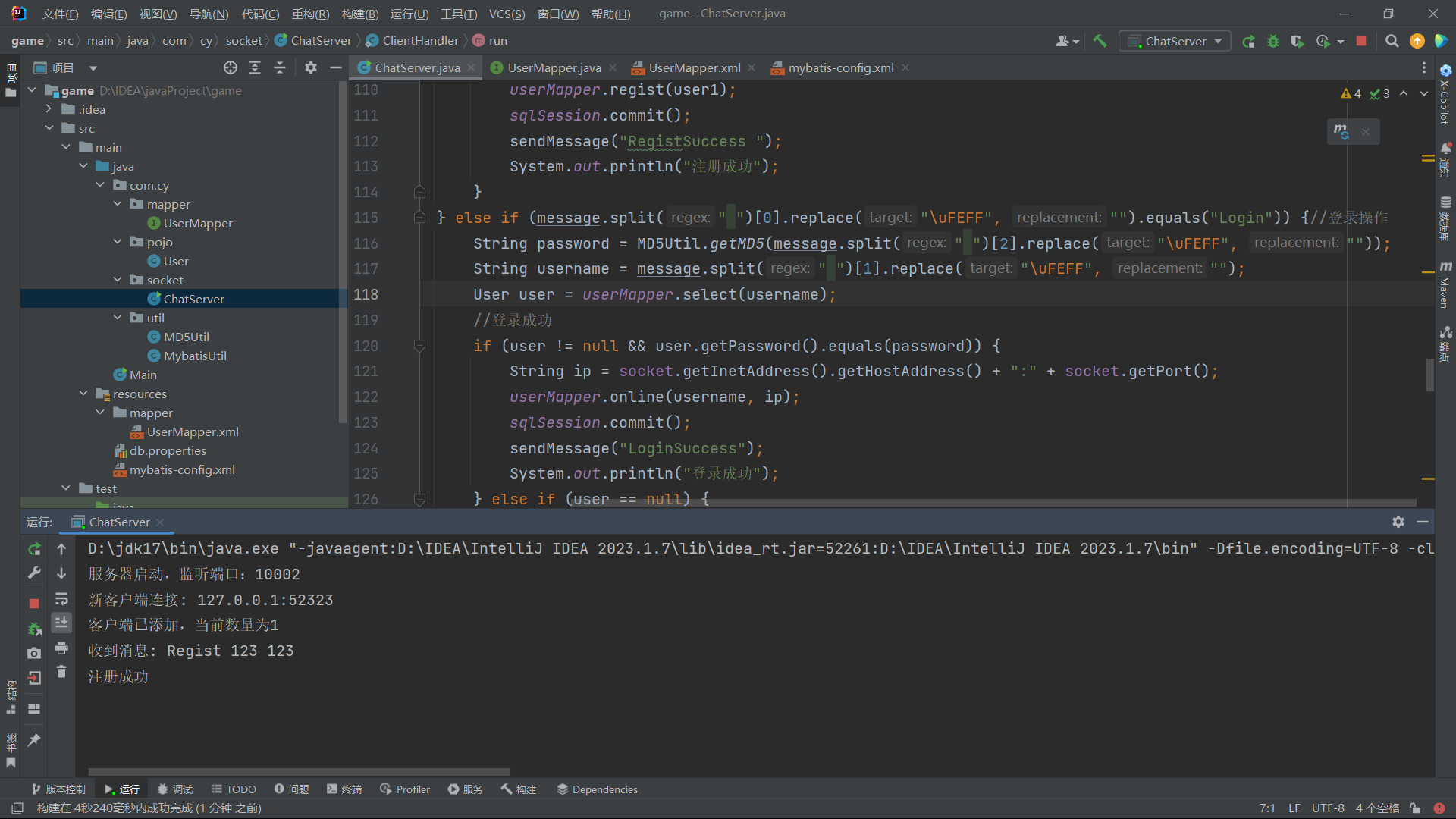
}

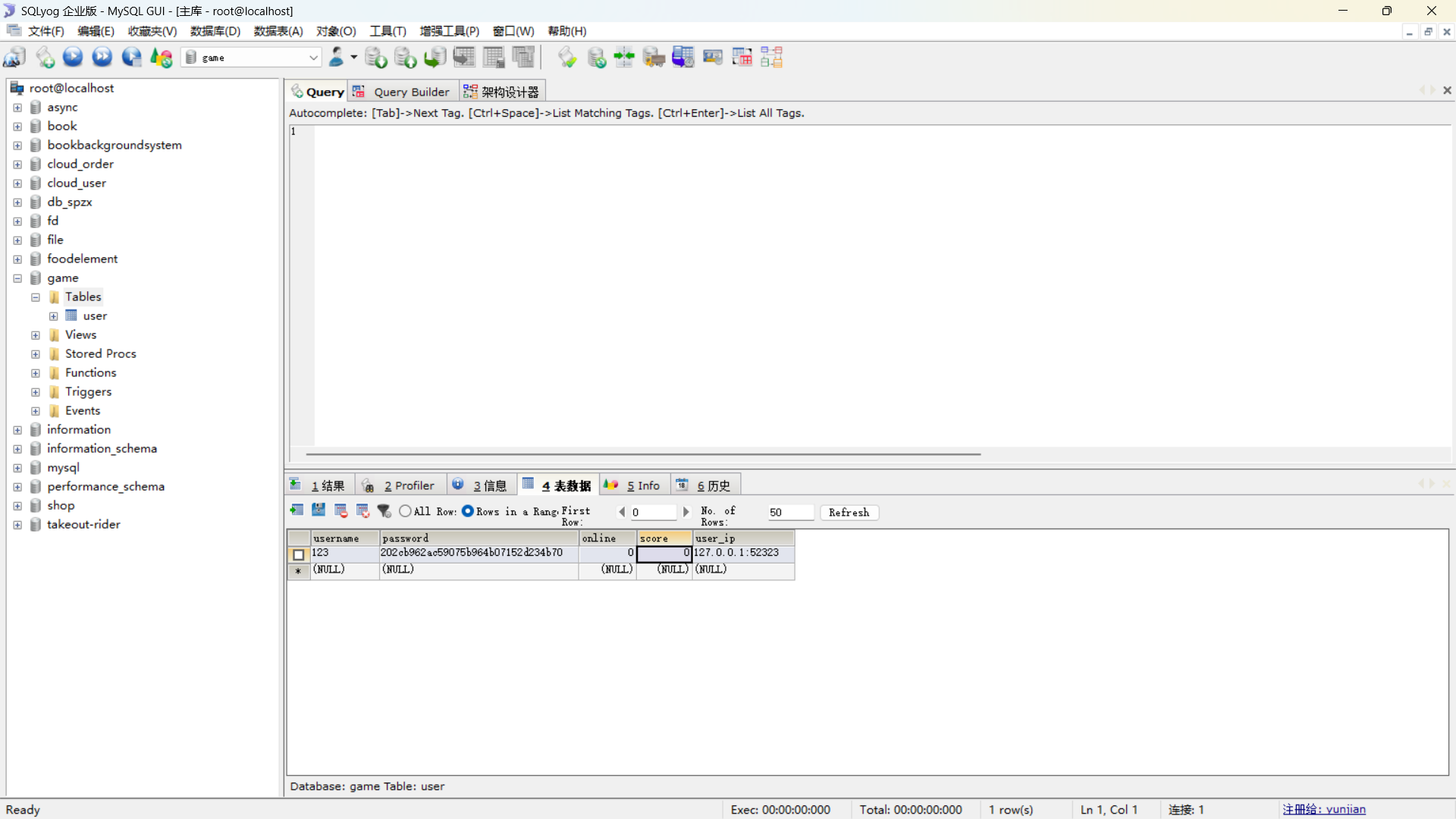
}

### 运行结果截图

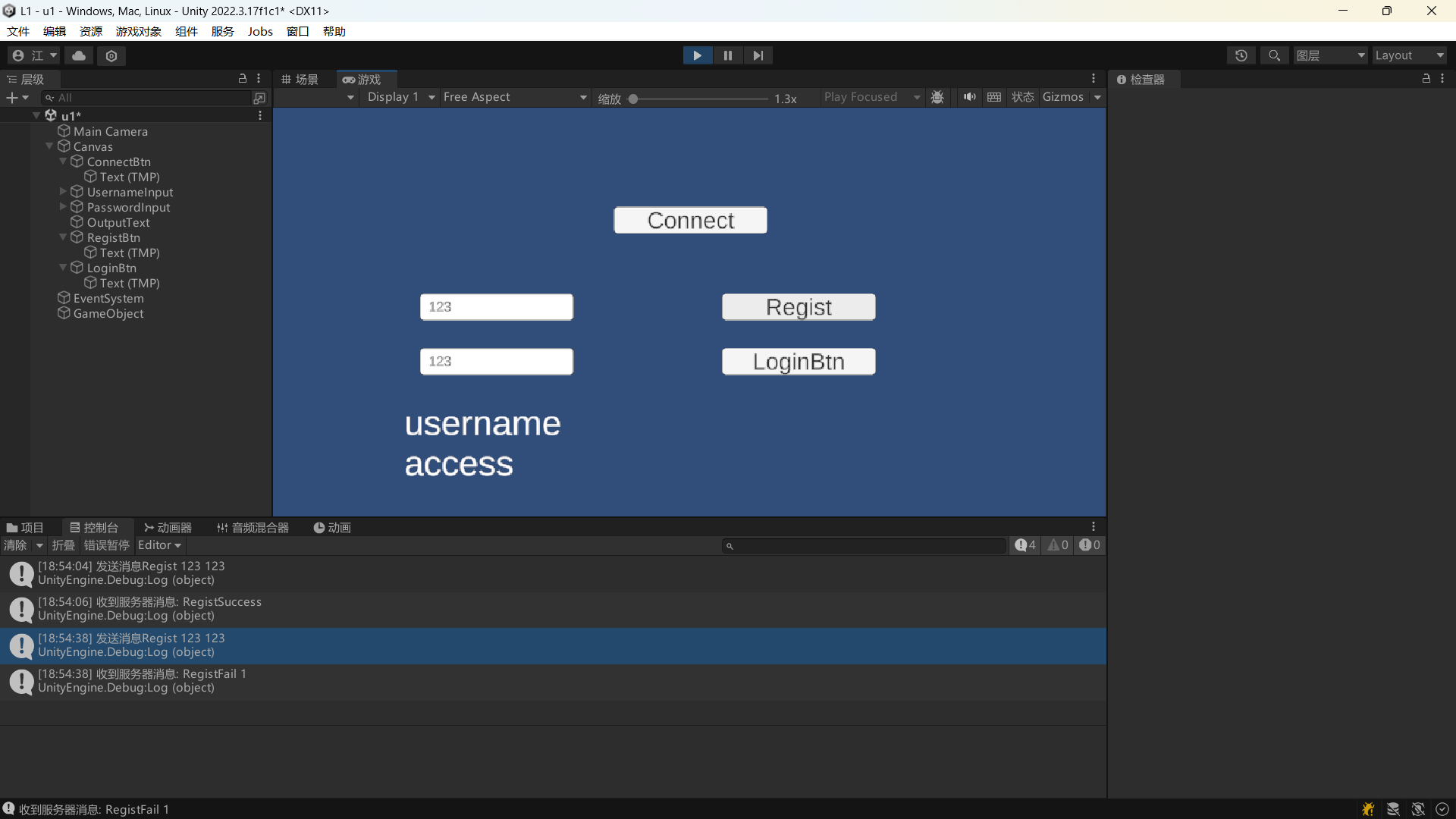
注册成功

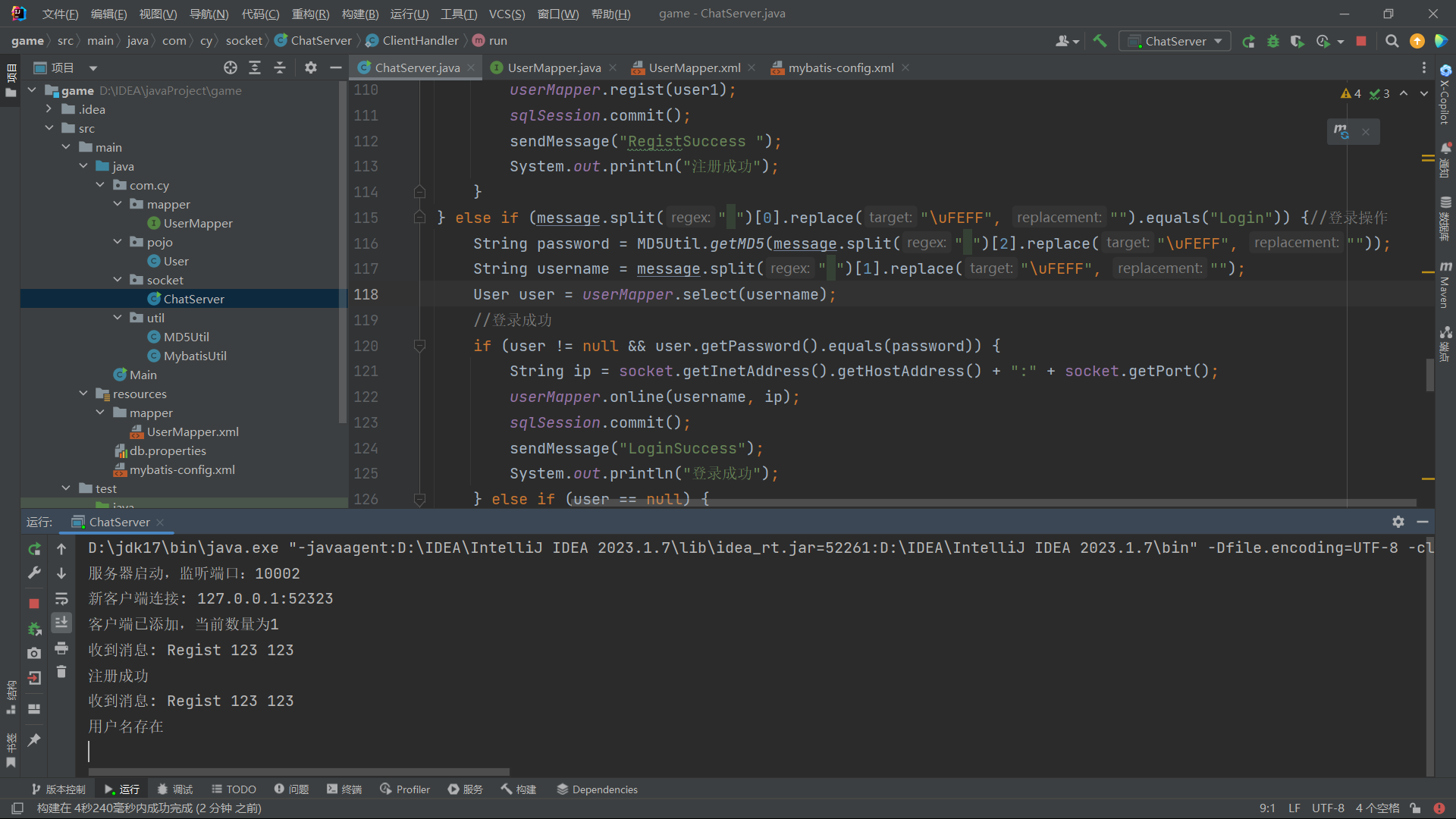




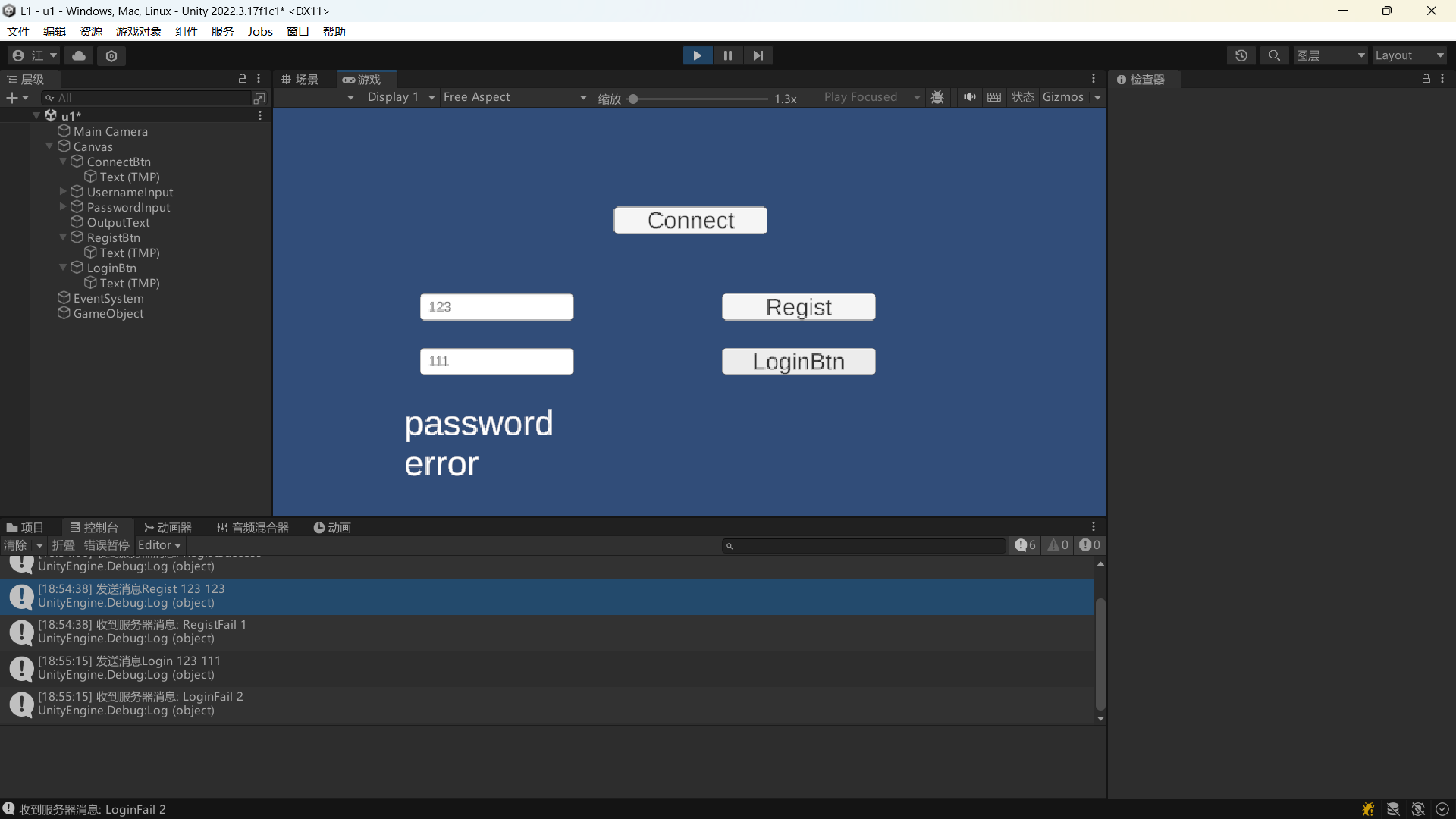


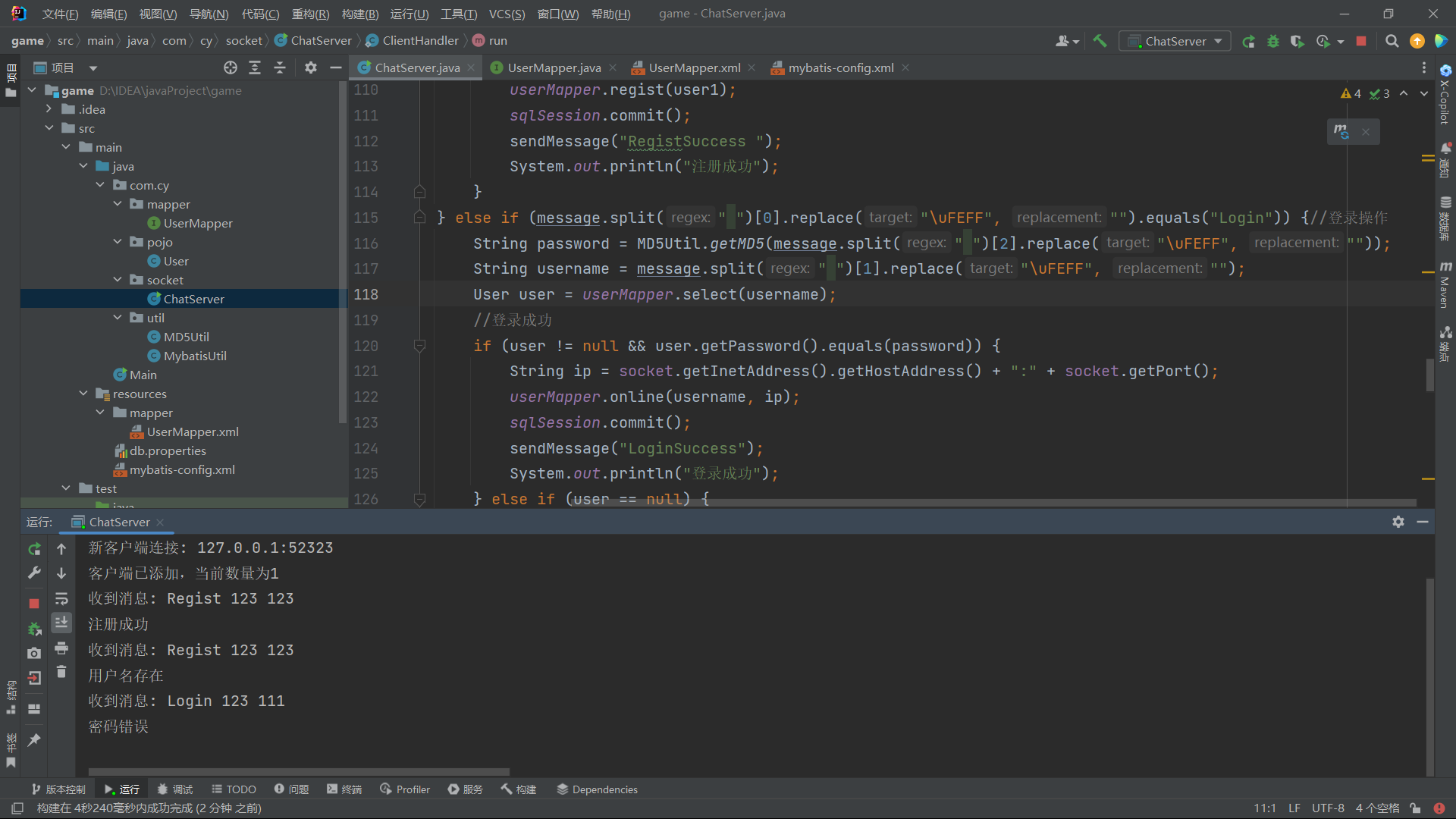
注册时，用户名已存在：



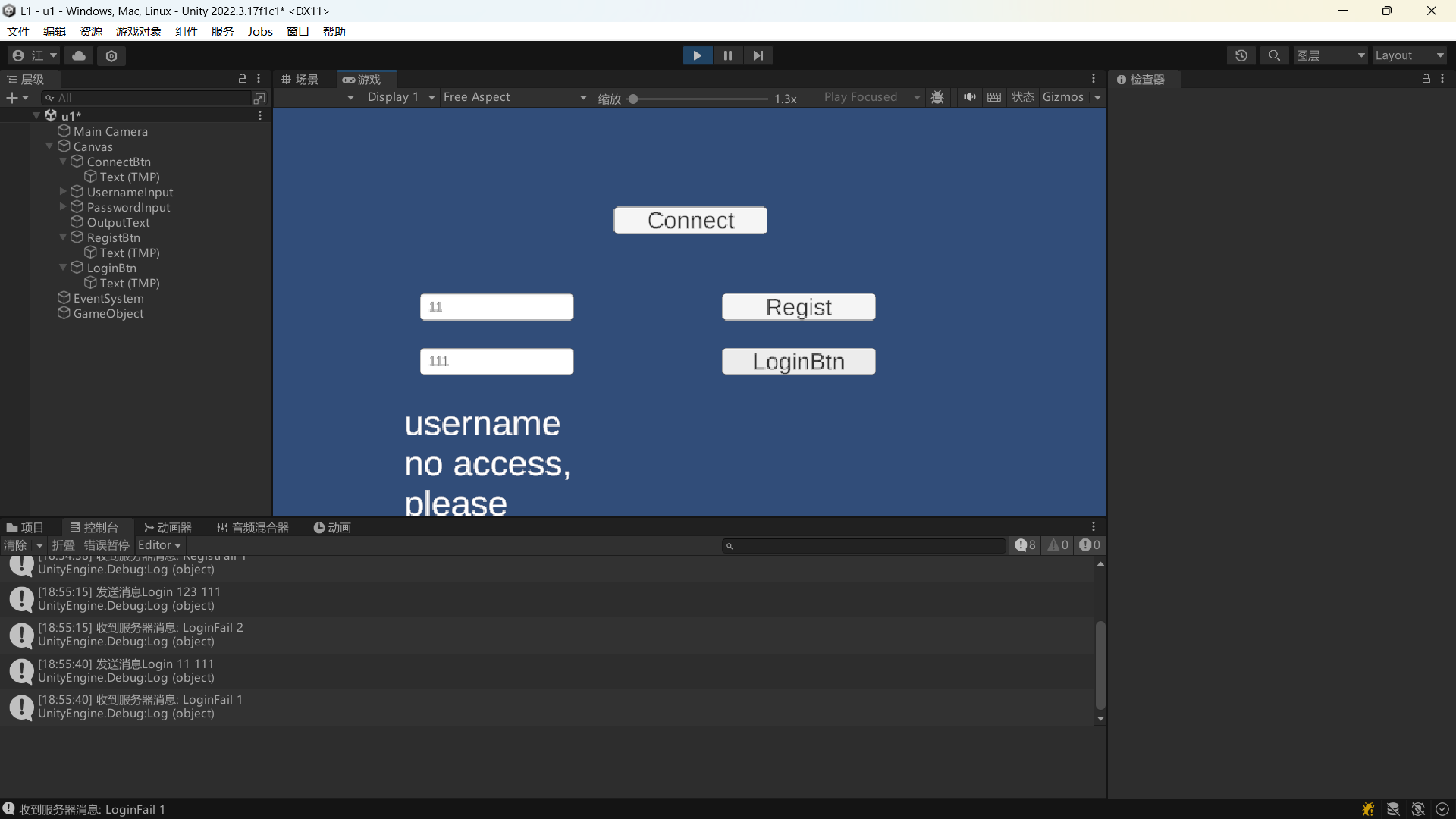


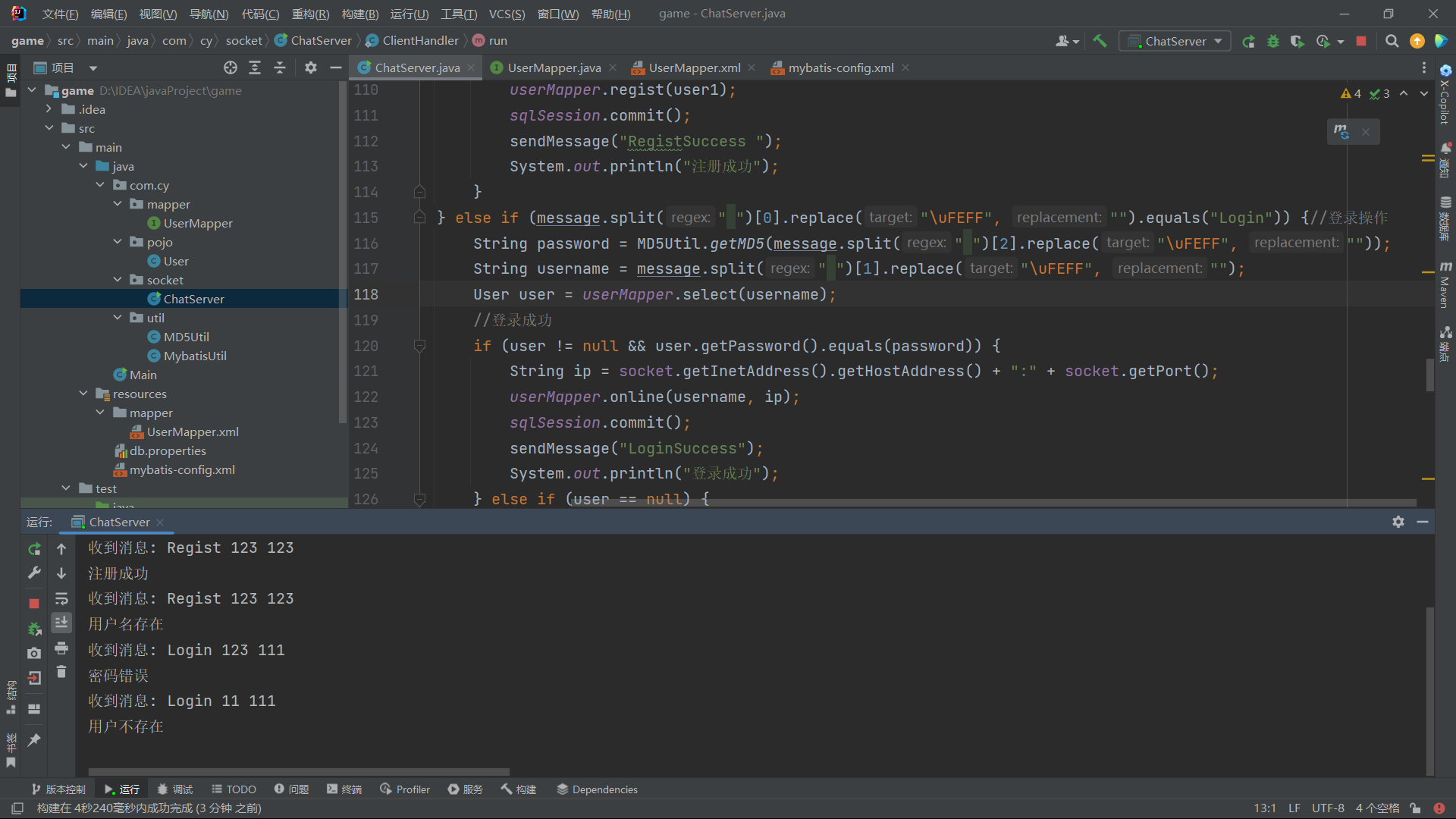
登录时密码错误：



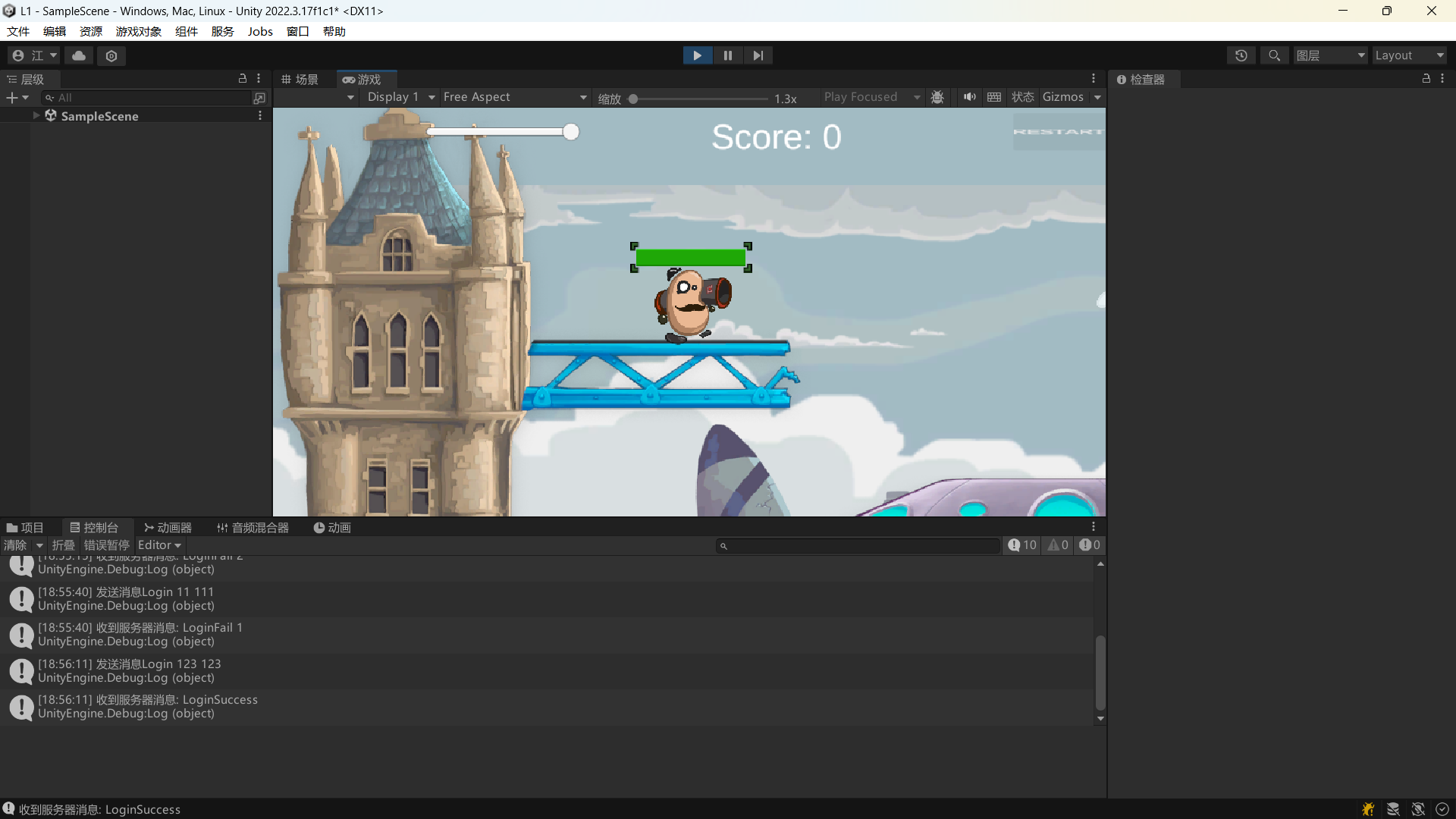


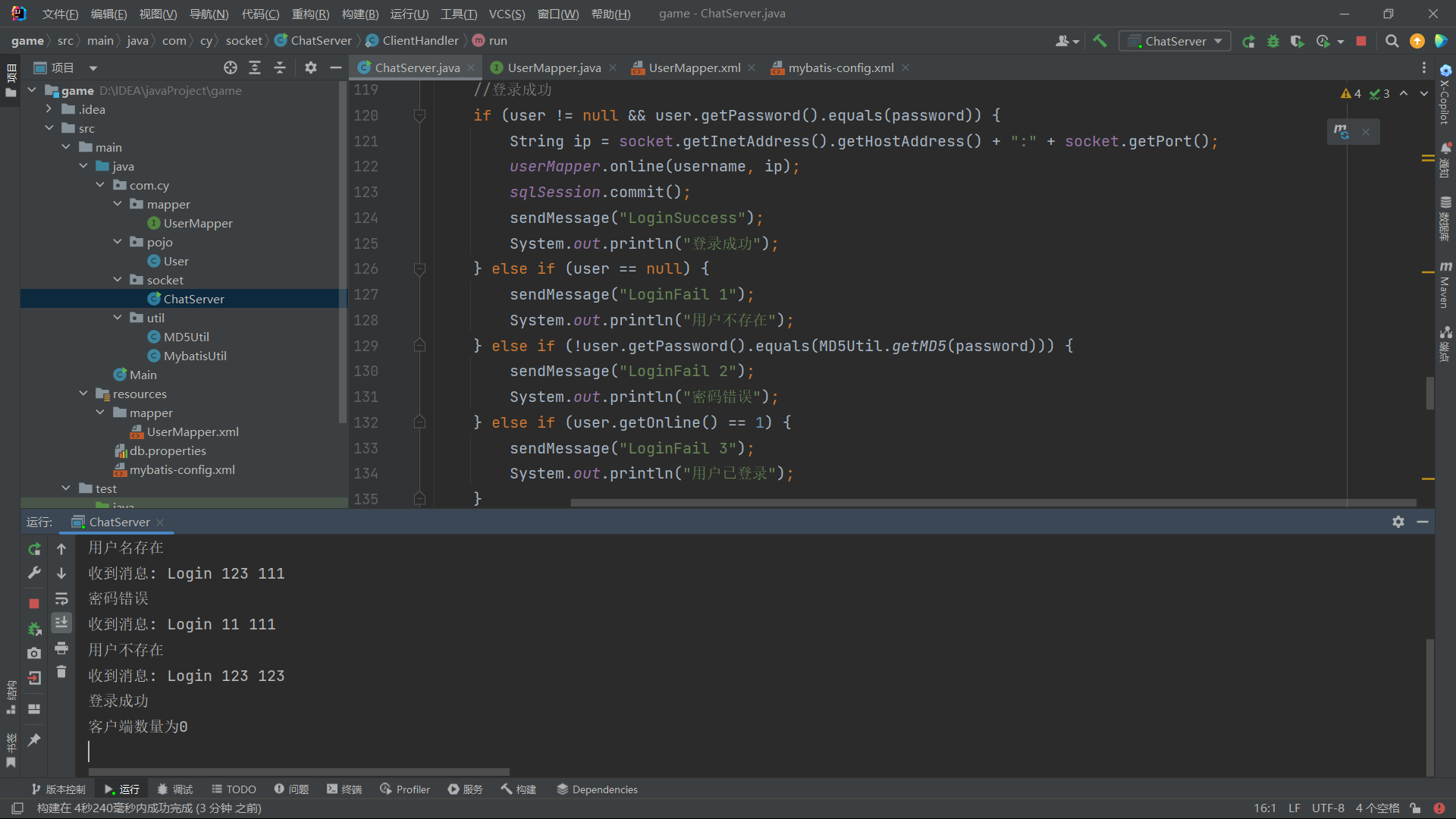
登录时用户名不存在：





登录成功：





## 三【上机实验中的其他它问题及心得】

这次实验我学会了如何注册登录与服务器和客户端的交互。