VD1

import java.net.\*;

import java.io.\*;

public class VD1Client {

    public static void main (String [] args ){

        try{

            //Tao mot socket toi dia chi ip va cong cua may chu

            Socket socket= new Socket("localhost",1234);

            //Mo luong vao ra de giao tiep voi may chu

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            PrintWriter out = new PrintWriter(socket.getOutputStream(),true);

            //Gui thong diep toi may chu

            String message = "Hello Server";

            out.println(message);

            //Doc phan hoi tu may chu

            String serverResponse = in.readLine();

            System.out.println("Server says: " + serverResponse);

            //dong ket noi

            socket.close();

        }catch (IOException e)

        {

            e.printStackTrace();

        }

    }

}

import java.io.\*;

import java.net.\*;

public class VD1Server {

    public static void main (String [] args){

    try{

        //Tao mot ServerSocket va rang buoc no voi cong 1234

        ServerSocket serverSocket = new ServerSocket(1234);

        System.out.println("Server is running and waiting for client connections...");

        //Chap nhan yeu cau ket noi tu mot may khach

        Socket clientSocket = serverSocket.accept();

        System.out.println("Client connected");

        //Mo luong vao ra de giao tiep voi may khach

        BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),true);

        //Doc du lieu tu may khach va gui phan hoi

        String clientMessage = in.readLine();

        System.out.println("Client says: " + clientMessage);

        out.println("Server received message: " + clientMessage);

        //Dong ket noi

        clientSocket.close();

        serverSocket.close();

    }catch (IOException e)

    {

        e.printStackTrace();

    }

 }

}

VD2

import java.util.HashMap;

import java.util.Map;

public class VD2 {

    public static void main (String [] args){

        //Tạo một đối tượng HashMap để lưu trữ các cập khóa-giá trị

        Map <String, Integer> studentGrades = new HashMap<>();

        //Thêm các cặp khóa-giá trị vào HashMap

        studentGrades.put("Alice",85);

        studentGrades.put("Bob", 92);

        studentGrades.put("Charlie", 78);

        studentGrades.put("Diana", 90);

        //Truy cập giá trị giữa dựa trên khóa

        int aliceGrade = studentGrades.get("Alice");

        System.out.println("Alice's grade: " + aliceGrade);

        //Xóa một cặp khóa-giá trị từ HashMap

        studentGrades.remove("Charlie");

        //Kiểm tra xem HashMap có chứa một khóa cụ thể hay không

        boolean containsBob =studentGrades.containsKey("Bob");

        System.out.println("Contains Bob? " + containsBob);

        //Lấy số lượng cặp khóa-giá trị trong HashMap

        int size = studentGrades.size();

        System.out.println("Number of students: " + size);

        //Lặp qua tất cả các cặp khóa-giá trị trong HashMap

        for (Map.Entry<String,Integer> entry : studentGrades.entrySet()){

            String studentName = entry.getKey();

            int grade = entry.getValue();

            System.out.println(studentName + " : " + grade);

        }

    }

}

Bài1

import java.io.\*;

import java.net.\*;

import java.util.HashMap;

public class Bai1Server {

    private static HashMap<String, String> accounts = new HashMap<>();

    public static void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(12345)) {

            System.out.println("Server đang chạy...");

            while (true) {

                Socket clientSocket = serverSocket.accept();

                new ClientHandler(clientSocket).start();

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    private static class ClientHandler extends Thread {

        private Socket clientSocket;

        public ClientHandler(Socket socket) {

            this.clientSocket = socket;

        }

        @Override

        public void run() {

            try (BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

                 PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {

                String request = in.readLine();

                String[] credentials = request.split(" ");

                String username = credentials[0];

                String password = credentials[1];

                synchronized (accounts) {

                    if (accounts.containsKey(username)) {

                        out.println("Error: Username đã tồn tại.");

                    } else if (password.length() < 8) {

                        out.println("Error: Mật khẩu phải dài hơn 8 ký tự.");

                    } else {

                        accounts.put(username, password);

                        out.println("Đăng ký thành công.");

                    }

                }

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

}

import java.io.\*;

import java.net.\*;

public class Bai1Client {

    public static void main(String[] args) {

        try (Socket socket = new Socket("localhost", 12345);

             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

             BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in))) {

            System.out.println("Nhập username:");

            String username = stdIn.readLine();

            System.out.println("Nhập password:");

            String password = stdIn.readLine();

            out.println(username + " " + password);

            String response = in.readLine();

            System.out.println("Server: " + response);

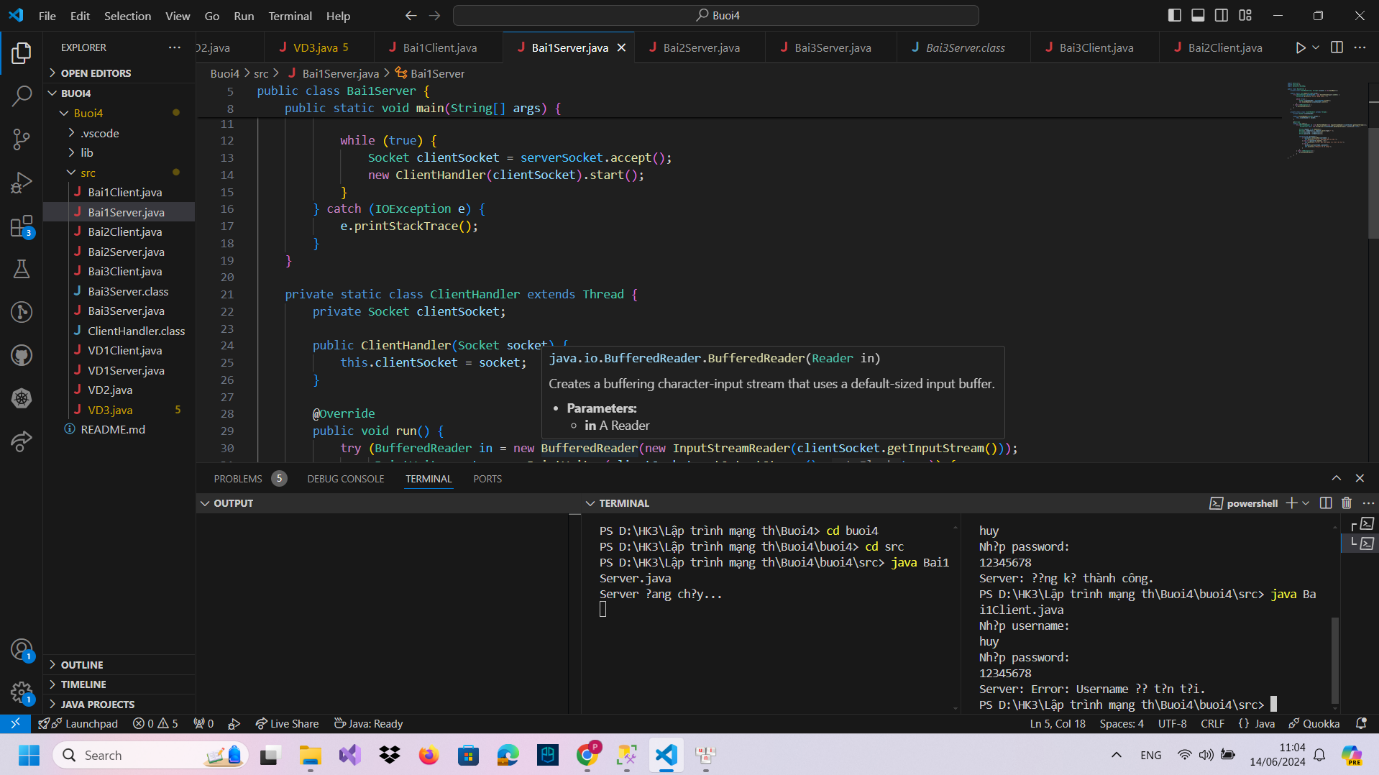
        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}



Bài 2

import java.io.\*;

import java.net.\*;

import java.util.Random;

public class Bai2Server {

    public static void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(12345)) {

            System.out.println("Server đang chạy...");

            while (true) {

                Socket clientSocket = serverSocket.accept();

                System.out.println("Client đã kết nối: " + clientSocket.getInetAddress());

                new ClientHandler(clientSocket).start();

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

class ClientHandler extends Thread {

    private Socket clientSocket;

    private int numberToGuess;

    private int attempts;

    public ClientHandler(Socket socket) {

        this.clientSocket = socket;

        Random random = new Random();

        this.numberToGuess = random.nextInt(101); // Tạo số ngẫu nhiên từ 0 đến 100

        this.attempts = 0;

    }

    @Override

    public void run() {

        try (BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream(), "UTF-8"));

             PrintWriter out = new PrintWriter(new OutputStreamWriter(clientSocket.getOutputStream(), "UTF-8"), true)) {

            String guess;

            out.println("Chào mừng đến với trò chơi đoán số! Hãy đoán số từ 0 đến 100:");

            while ((guess = in.readLine()) != null) {

                attempts++;

                int guessedNumber;

                try {

                    guessedNumber = Integer.parseInt(guess);

                } catch (NumberFormatException e) {

                    out.println("Vui lòng nhập một số nguyên.");

                    continue;

                }

                if (guessedNumber < numberToGuess) {

                    out.println("Số bạn đoán nhỏ hơn số cần tìm. Hãy đoán lại:");

                } else if (guessedNumber > numberToGuess) {

                    out.println("Số bạn đoán lớn hơn số cần tìm. Hãy đoán lại:");

                } else {

                    out.println("Chính xác! Bạn đã đoán đúng số sau " + attempts + " lần.");

                    break;

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        } finally {

            try {

                clientSocket.close();

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

}

import java.io.\*;

import java.net.\*;

public class Bai2Client {

    public static void main(String[] args) {

        try (Socket socket = new Socket("localhost", 12345);

             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream(), "UTF-8"));

             PrintWriter out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), "UTF-8"), true);

             BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in, "UTF-8"))) {

            System.out.println("Đã kết nối đến server.");

            String serverResponse;

            while ((serverResponse = in.readLine()) != null) {

                System.out.println("Server: " + serverResponse);

                if (serverResponse.contains("Chính xác")) {

                    break;

                }

                System.out.print("Nhập số bạn đoán: ");

                String userGuess = stdIn.readLine();

                out.println(userGuess);

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

A screenshot of a computer

Description automatically generated

Bài 3

import java.io.\*;

import java.net.\*;

import java.sql.\*;

import java.util.\*;

public class Server {

    private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=ltm;integratedSecurity=true;";

    // Phương thức getter cho biến URL

    public static String getURL() {

        return URL;

    }

    public static void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(5000)) {

            System.out.println("Server is listening on port 5000");

            while (true) {

                new ClientHandler(serverSocket.accept()).start();

            }

        } catch (IOException ex) {

            ex.printStackTrace();

        }

    }

}

class ClientHandler extends Thread {

    private Socket socket;

    public ClientHandler(Socket socket) {

        this.socket = socket;

    }

    public void run() {

        try {

            // Đăng ký JDBC Driver (thường không cần thiết với các phiên bản JDBC hiện đại)

            try {

                Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

            } catch (ClassNotFoundException e) {

                e.printStackTrace();

                return;

            }

            // Kết nối tới cơ sở dữ liệu SQL Server sử dụng getter cho URL

            try (Connection connection = DriverManager.getConnection(Server.getURL());

                 BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

                 PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

                // Đọc tên tác giả gửi từ client

                String authorName = in.readLine();

                List<String> results = searchAuthor(connection, authorName);

                // Gửi kết quả tìm kiếm về client

                if (results.isEmpty()) {

                    out.println("No authors found");

                } else {

                    out.println("Authors found:");

                    results.forEach(out::println);

                }

            } catch (SQLException ex) {

                ex.printStackTrace();

            }

        } catch (IOException ex) {

            ex.printStackTrace();

        }

    }

    // Phương thức tìm kiếm tác giả trong cơ sở dữ liệu

    private List<String> searchAuthor(Connection connection, String authorName) throws SQLException {

        List<String> authors = new ArrayList<>();

        String query = "SELECT first\_name, last\_name, email, birthdate FROM authors WHERE first\_name = ? OR last\_name = ?";

        try (PreparedStatement statement = connection.prepareStatement(query)) {

            statement.setString(1, authorName);

            statement.setString(2, authorName);

            try (ResultSet resultSet = statement.executeQuery()) {

                while (resultSet.next()) {

                    String author = String.format("%s %s, %s, %s",

                            resultSet.getString("first\_name"),

                            resultSet.getString("last\_name"),

                            resultSet.getString("email"),

                            resultSet.getString("birthdate"));

                    authors.add(author);

                }

            }

        }

        return authors;

    }

}

import java.io.\*;

import java.net.\*;

public class Client {

    public static void main(String[] args) {

        try (Socket socket = new Socket("localhost", 5000);

             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

             BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in))) {

            System.out.println("Enter author name to search: ");

            String authorName = stdIn.readLine();

            out.println(authorName);

            String response;

            while ((response = in.readLine()) != null) {

                System.out.println(response);

            }

        } catch (IOException ex) {

            ex.printStackTrace();

        }

    }

}

