TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA Công Nghệ Thông Tin
BỘ MÔN: MẠNG & TT

ĐỀ THI VÀ BÀI LÀM

Tên học phần: Lập Trình Mạng
Mã học phần: Hình thức thi: *Tự luận có giám sát*
Thời gian làm bài: 60 phút *(không kể thời gian phát đề và nộp bài)*
- **Không được** sử dụng lài liệu trên internet khi làm bài, nhưng được sử dụng tài liệu có sẵn trên máy tính.
- **Không** chia sẻ bài cho nhau cũng như sử dụng AI để làm bài.

Họ tên:…Cáp Kim Khánh…**Lớp**:…23T_DT1…**MSSV**:…102230192

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV_HọTên.pdf và nộp bài thông qua MSTeam:

Câu 1 (*6 điểm*): Hãy viết chương trình theo giao thức TCP với các chức năng sau:

I. Client (xem yêu cầu a. ii. để thiết kế giao diện cho phù hợp):
+ Kết nối và gửi tới server 1 chuỗi "**Please livestream**":
  a. Nếu nhận được chuỗi "**OK**" thì lặp đi lặp lại các bước sau:
    i. Gửi cho server chuỗi "**Get Video**"
    ii. Nhận từ server mảng byte chứa thông tin hình ảnh và dùng Graphics để hiển thị hình ảnh đã nhận.
  b. Nếu không nhận được chuỗi "**OK**" hoặc quá 10s không thấy phản hồi thì kết thúc chương trình.

II. Server:
+ Tạo 1 danh sách tĩnh các client không cho phép kết nối.
  Ví dụ: BlackList:["192.168.10.100", :["192.168.10.105"]
+ Lắng nghe và chấp nhận kết nối từ nhiều client nếu IP không thuộc các địa chỉ không cho phép.
+ Nhận từ client đã kết nối 1 chuỗi, kiểm tra chuỗi đó có phải là:
  a. Nếu là "**Please livestream**" thì phản hồi cho client chuỗi "**OK**".
  b. Nếu là "**Get Video**" thì gửi gửi mảng byte chứa thông tin hình ảnh màn hình server cho client.

Lưu ý:

1. Tối ưu quá trình livestream để không phải chụp màn hình nhiều lần nếu có nhiều client kết nối tới.
2. Nếu client ngắt kết nối đột ngột cần xử lý loại bỏ kết nối để không phải lưu trữ thông tin client đó.

 # Trả lời:
Dán code server vào bên dưới

```
package THICK;

import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;

import java.net.ServerSocket;
import java.net.Socket;

import java.util.Vector;

import javax.imageio.ImageIO;
```

```java
public class server {
    private static Vector<String> blackList = new Vector<>(); // BlackList save IP client
    public static final Object lock = new Object();

    public static void main(String[] args) {
        new server();
    }

    public server() {
        Screen s = new Screen();

        s.start();
        try {
            ServerSocket server = new ServerSocket(2345);
            while (true) {
                Socket soc = server.accept();
                String clientIP = soc.getInetAddress().getHostAddress();

                //Test band
                //blackList.add(clientIP);

                if (blackList.contains(clientIP)) {
                    System.out.println("Client ID: " + soc.getInetAddress().getHostAddress() + " is Band!");

                    soc.close();
                    continue;
                }

                ScreenProcessing sp = new ScreenProcessing(soc);
                sp.start();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}

class Screen extends Thread {
    static byte[] tmp;

    public void run() {
        Robot r = null;
        Rectangle capture = null;

        try {
            r = new Robot();
            capture = new Rectangle(Toolkit.getDefaultToolkit().getScreenSize());
        } catch (Exception e) {
            e.printStackTrace();
        }

        while (true) {
            try {
                BufferedImage img = r.createScreenCapture(capture);
```

```
            ByteArrayOutputStream bos = new ByteArrayOutputStream();

            ImageIO.write(img, "png", bos);
            bos.flush();

            synchronized (server.lock) {
                tmp = bos.toByteArray();
            }

            Thread.sleep(20);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

class ScreenProcessing extends Thread {
    Socket soc;

    public ScreenProcessing(Socket soc) {
        this.soc = soc;
    }

    public void run() {
        DataInputStream dis = null;
        DataOutputStream dos = null;
        try {
            dis = new DataInputStream(soc.getInputStream());
            dos = new DataOutputStream(soc.getOutputStream());

            String msg = dis.readUTF();
            System.out.println("Server received: " + msg);

            if (msg.equalsIgnoreCase("Please livestream")) {
                dos.writeUTF("OK");
                dos.flush();

                System.out.println("Server Send: OK");

                while (true) {
                    String request = dis.readUTF();
                    if (request.equalsIgnoreCase("Get Video")) {
                        byte[] data = null;
                        synchronized (server.lock) {
                            data = Screen.tmp;
                        }

                        if (data != null) {
                            dos.writeInt(data.length);
                            dos.write(data);
                            dos.flush();
                        } else {
                            // Send 0 if no data yet
                            dos.writeInt(0);
```

```
                    dos.flush();
                }
            } else {
                // Disconnect
                break;
            }
        }
    }

} catch (Exception e) {
    // Client disconnected
    System.out.println("Client is Disconnected");
} finally {
    try {
        if (soc != null)
            soc.close();
    } catch (Exception ex) {
    }
}
}
}
```

Dán code client vào bên dưới

```
package THICK;

import java.awt.Graphics;
import java.awt.image.BufferedImage;

import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;

import java.net.Socket;

import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class client extends JFrame {
    Socket soc;
    BufferedImage currentImage = null;
    boolean isRunning = true;
    JPanel screenPanel;

    public static void main(String[] args) {
        new client();
    }

    public client() {
        this.setTitle("Share Screen");
        this.setSize(800, 600);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        screenPanel = new JPanel() {
```

```
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            if (currentImage != null) {
                g.drawImage(currentImage, 0, 0, this.getWidth(), this.getHeight(), null);
            }
        }
    };
    this.add(screenPanel);
    this.setVisible(true);

    new Thread(() -> {
        try {
            soc = new Socket("localhost", 2345);
            soc.setSoTimeout(10000); // 10s timeout

            DataOutputStream dos = new DataOutputStream(soc.getOutputStream());
            DataInputStream dis = new DataInputStream(soc.getInputStream());

            dos.writeUTF("Please livestream");
            System.out.println("Client Send: PLEASE LIVESTREAM");
            dos.flush();

            String response = dis.readUTF();
            System.out.println("Client received: " + response);
            if ("OK".equalsIgnoreCase(response)) {
                while (isRunning) {
                    try {
                        dos.writeUTF("Get Video");
                        System.out.println("Client Send: GET VIDEO");
                        dos.flush();

                        int len = dis.readInt();
                        if (len > 0) {
                            byte[] data = new byte[len];
                            dis.readFully(data);

                            ByteArrayInputStream bais = new ByteArrayInputStream(data);
                            currentImage = ImageIO.read(bais);
                            screenPanel.repaint();
                        }

                        Thread.sleep(50);
                    } catch (Exception e) {
                        System.out.println("Error during streaming: " + e.getMessage());
                        break;
                    }
                }
            } else {
                System.out.println("Server rejected request (BAND)!");
            }
        } catch (Exception e) {
            System.out.println("Connection error or timeout: " + e.getMessage());
        } finally {
            try {
```

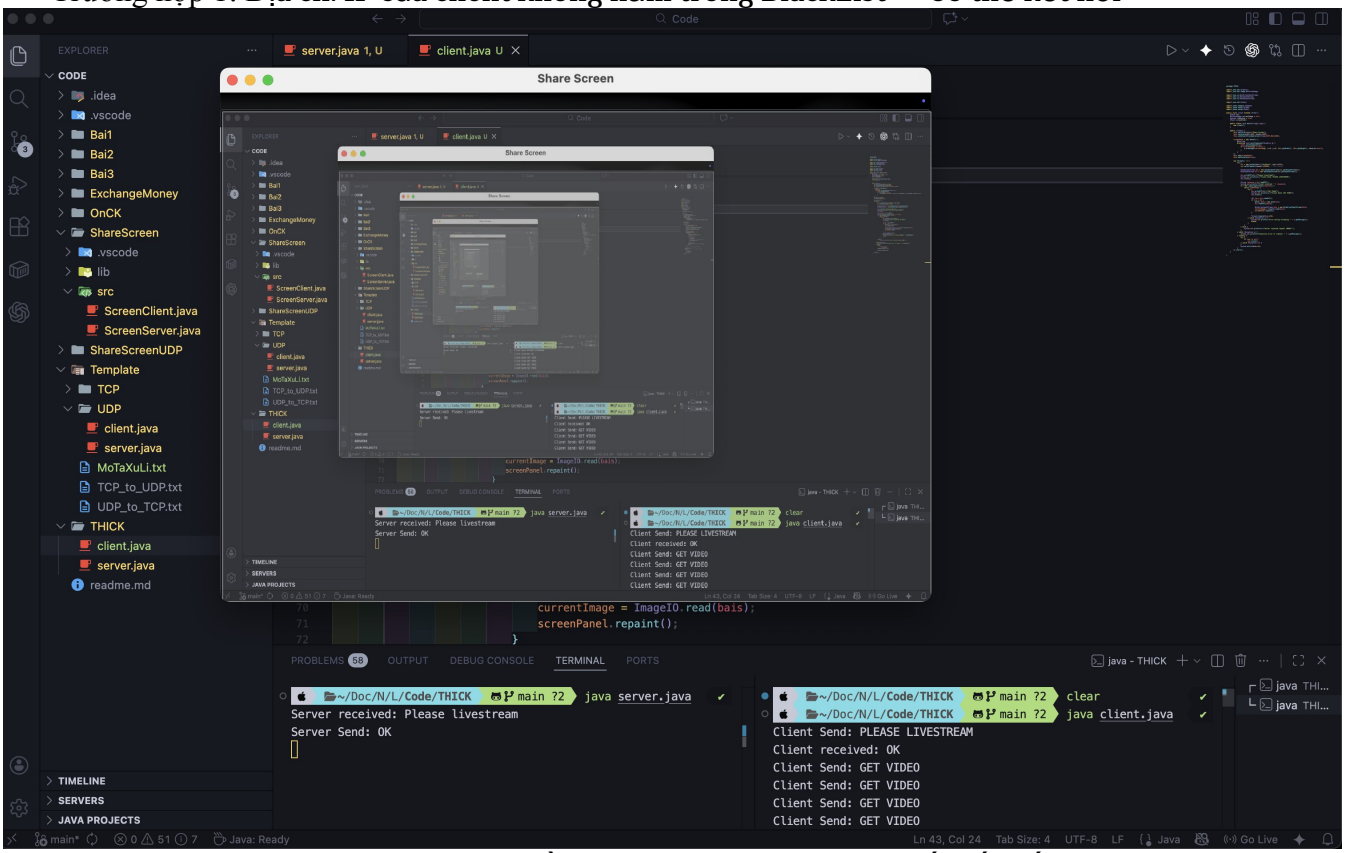
```
                if (soc != null)
                    soc.close();
                } catch (Exception ex) {
                }
                System.exit(0);
            }
        }).start();
    }

}
```

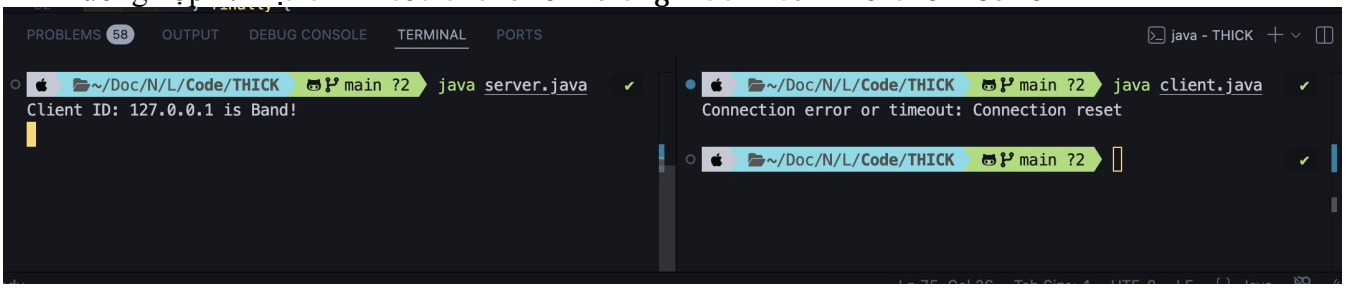
Dán các kết quả thực thi vào bên dưới

(Chú ý phải hiển thị các thông điệp giữa client và server trong quá trình thực thi. Thực thi tất cả các trường hợp có thể có!!)

- Trường hợp 1: Địa chỉ IP của client không nằm trong BlackList -> Có thể kết nối



- Trường hợp 2: Địa chỉ IP của client nằm trong BlackList -> Từ chối kết nối



Câu 2 (*4 điểm*): Hãy cho biết nếu chuyển chương trình ở câu 1 qua giao thức UDP thì các công việc cần làm gồm những gì? Nêu ưu nhược điểm khi sử dụng UDP, tại sao? (Không cần code, chỉ cần mô tả)

 # Trả lời:

Trong giao thức TCP, ta dùng luồng (DataInputStream & DataOutputStream) để truyền tin cậy trên một "đường ống" bằng kết nối bắt tay 3 chiều. Trong khi đó, chuyển sang UDP, ta phải xử lý theo dạng phi kết nối, sử dụng DatagramSocket để làm cổng giao tiếp và DatagramPacket để đóng gói dữ liệu rời rạc.

- Khởi tạo DatagramSocket ở cả hai phía (Server gắn với port cố định, Client tạo socket ngẫu nhiên hoặc cố định) để thiết lập điểm gửi/nhận.
- Chuyển đổi dữ liệu (chuỗi, số) thành mảng byte (byte[]) và tạo DatagramPacket chứa mảng này kèm địa chỉ IP/Port đích để gửi đi.
- Tạo DatagramPacket chứa mảng byte rỗng (buffer) với kích thước định trước để làm vùng đệm hứng dữ liệu.
- Sử dụng phương thức DatagramSocket.send() để đẩy gói tin đi và DatagramSocket.receive() để chờ và nhận gói tin về.

Ưu điểm của UDP:

- Rất nhanh, không cần phải xác minh kết nối (không có cơ chế bắt tay 3 chiều), độ trễ thấp.
- Dữ liệu được truyền đi ngay lập tức, không cần chờ phản hồi ACK.

Ngược điểm của UDP:

- Mất gói tin, độ tin cậy thấp.
- Sai thứ tự các gói tin.
- Không kiểm soát tắc nghẽn vì UDP liên tục gửi dữ liệu đi mặc dù mạng đang bị tắt nghẽn, dễ dẫn đến các gói packet mất hàng loạt.

Ký duyệt của trưởng bộ môn

Đà Nẵng, ngày 12 tháng 12 năm 2025