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

ĐỀ THI VÀ BÀI LÀM - Đề 1

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)*
Được sử dụng tài liệu khi làm bài. Không chia sẻ bài cho nhau, nếu phát hiện sẽ chia đều số điểm.

Họ tên:…Nguyễn Huynh……**Lớp**:…22T_KHDL……**MSSV**:……102220024……...

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 (*4 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 LTM;

import java.awt.*;

import java.awt.image.BufferedImage;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.Arrays;

import javax.imageio.ImageIO;
import java.awt.Robot;
import java.awt.Toolkit;
```

```
public class ServerVideo {
	private static final ArrayList<String> BLACKLIST = new
ArrayList<String>(Arrays.asList("192.168.10.100", "192.168.10.105"));

	public static void main(String[] args) throws Exception{
		ServerSocket server = new ServerSocket(7777);
		while(true) {
			Socket soc = server.accept();
			String clientIP = soc.getInetAddress().getHostAddress();

	if (BLACKLIST.contains(clientIP)) {
	   soc.close();
	   continue;
	}
			Test tmp = new Test(soc);
			tmp.start();
		}
	}

}

class Test extends Thread{
	Socket soc;
	DataInputStream dis;
	DataOutputStream dos;
	public Test(Socket soc) {
		this.soc = soc;
		try {
			this.dis = new DataInputStream(soc.getInputStream());
			this.dos = new DataOutputStream(soc.getOutputStream());
		}catch (Exception e) {
			e.printStackTrace();
		}
	}
	public void run() {
		try {
			String message = dis.readUTF();
			if(message.equals("Please livestream")) {
				dos.writeUTF("OK");
				String message1 = dis.readUTF();
				if(message1.equals("Get Video")) {
					Robot robot = new Robot();
				Rectangle screenRect = new
Rectangle(Toolkit.getDefaultToolkit().getScreenSize());
				BufferedImage screenShot = robot.createScreenCapture(screenRect);
		ByteArrayOutputStream baos = new ByteArrayOutputStream();
		ImageIO.write(screenShot, "png", baos);
		byte[] imageBytes = baos.toByteArray();

		dos.writeInt(imageBytes.length);
		dos.write(imageBytes);
		dos.flush();
				}
```

```
                }

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

        }
}
```

Dán code client vào bên dưới

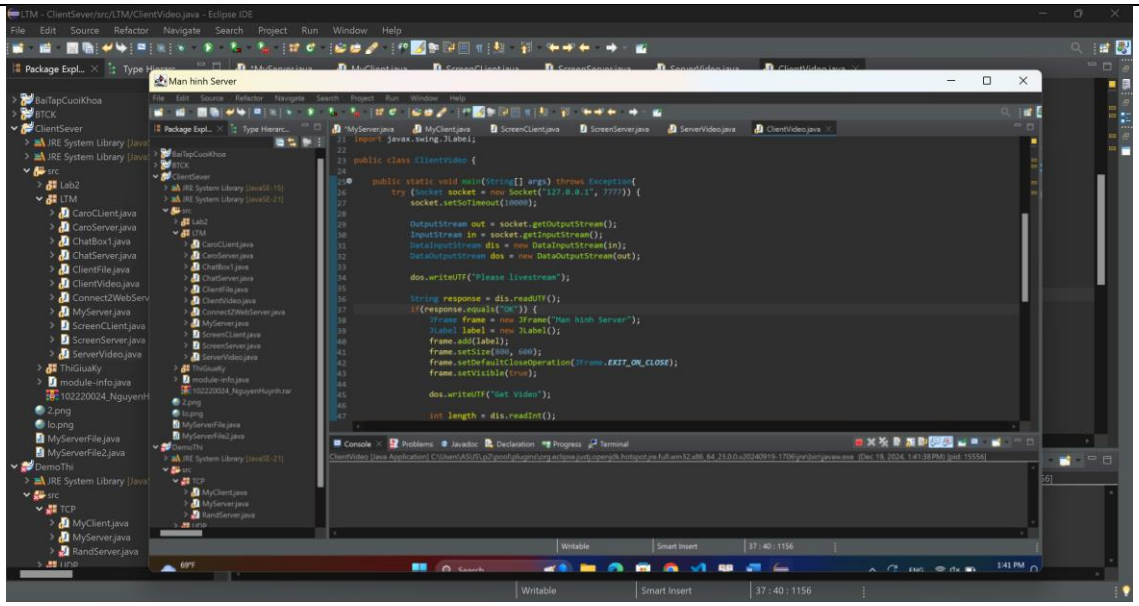
```
package LTM;

import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.SocketTimeoutException;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class ClientVideo {

    public static void main(String[] args) throws Exception{
        try (Socket socket = new Socket("127.0.0.1", 7777)) {
    socket.setSoTimeout(10000);

    OutputStream out = socket.getOutputStream();
    InputStream in = socket.getInputStream();
    DataInputStream dis = new DataInputStream(in);
    DataOutputStream dos = new DataOutputStream(out);

    dos.writeUTF("Please livestream");

    String response = dis.readUTF();
    if(response.equals("OK")) {
        JFrame frame = new JFrame("Man hinh Server");
       JLabel label = new JLabel();
       frame.add(label);
       frame.setSize(800, 600);
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       frame.setVisible(true);
```

```
            dos.writeUTF("Get Video");

            int length = dis.readInt();
            byte[] imageBytes = new byte[length];
            dis.readFully(imageBytes);

            ByteArrayInputStream bais = new ByteArrayInputStream(imageBytes);
            BufferedImage image = ImageIO.read(bais);

            if (image != null) {
               int imgWidth = image.getWidth();
               int imgHeight = image.getHeight();

               int newWidth = imgWidth * 4/5;
               int newHeight = imgHeight * 4/5;

               Image scaledImage = image.getScaledInstance(newWidth, newHeight,
Image.SCALE_SMOOTH);

               ImageIcon icon = new ImageIcon(scaledImage);
               label.setIcon(icon);

               frame.setSize(newWidth, newHeight);

               frame.repaint();
            }
         }
      } catch (SocketTimeoutException e) {
         System.out.println("Connection timed out.");
      } catch (IOException e) {
         e.printStackTrace();
      }
         }

}
```

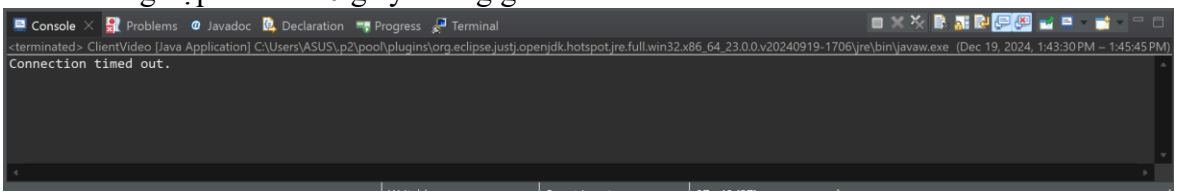
Dán các kết quả thực thi vào bên dưới

(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:Gửi và nhận hình ảnh thành công



- Trường hợp 2: Sau 10 giây không gửi hình ảnh



Câu 2 (*1 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:

- Về phần Logic xử lý hình ảnh thì giữ nguyên.
- Server:
  + Thay vì sử dụng ServerSocket để lắng nghe kết nối thì ta sử dụng DatagramSocket để nhận và gửi gói tin cho client.
  + Sử dụng DatagramPacket để tạo lập gói tin vơi các thông tin như nội dung gói tin, độ dài gói tin, địa chỉ ip và Port của client nhận gói tin.
- Client:
  + Thay vì dùng socket để kết nối với server rồi dùng DataInputStream và DataoutputStream để gửi và nhận thông tin thì ta dùng DatagramSocket để nhận và gửi gói tin và dùng DatagramPacket để tạo lập gói tin.

Câu 3 (5 *điểm*): Trong phần bài tập JSP/Servlet đã nộp

Đề bài tập: *Có 1 tính toán lớn (chạy ngầm, ví dụ như xử lý dữ liệu lớn, xử lý video, convert pdf qua doc, xây dựng mô hình học máy, kiểm tra đạo code, đạo văn tự động,...). Khi client gửi thông tin cần thực hiện xử lý, server sẽ đẩy thông tin đó vào 1 hằng đợi để thực hiện. Client sẽ xem kết quả xử lý thông qua account của bản thân*.

Hãy trả lời các câu hỏi sau:

a) Cho biết tên của các thành viên trong nhóm, kể cả bản thân

 # Trả lời: viết câu trả lời vào bên dưới

Thành viện:

- Nguyễn Huynh
- Hoàng Đức Mạnh
- Nguyễn Đức Hoài Vũ

b) Mô tả chức năng chính mà bản thân đã đóng góp vào trong chương trình.

 # Trả lời: viết câu trả lời vào bên dưới

- Giao diện hệ thống: Em đã code toàn bộ giao diện của hệ thống.
- Chức năng Login: người dùng đăng nhập vào hệ thống bằng Username và Password. Nếu người dùng đăng nhập thành công thì dùng Session để lưu lại UserID và Username còn nếu đăng nhập thất bại thì trở lại trang đăng nhập.
- Chức năng đăng ký: người dùng nhập đầy đủ thông tin Username và Password để đăng ký tài khoản. Hệ thống sẽ kiểm tra xem Username tồn tại hay chưa nếu chưa thì tạo tài khoản lên cở sở dữ liệu và chuyển sang trang giao diện màn hình chính. Còn nếu thất bại thì thông báo "User already exists or invalid input." Và quay trở lại giao diện đăng ký.
- Chức năng convert pdf to doc: Người dùng chọn file pdf cho hệ thống chuyển đổi. Hệ thống chia file pdf ban đầu thành các file pdf nhỏ hơn (bằng thư viện pdfbox) với kích thước <= 10 trang. Chuyển từng file pdf nhỏ thành doc (spire.pdf) rồi nối các file doc lại với nhau (spire.doc) tạo thành file output hoàn chỉnh. Khi file được chuyển đổi xong thì trạng thái chuyển thành “Success” và người dùng có thể tải về được. Lý do chia nhỏ 10 trang vì thư viện spire.pdf chỉ cho phép chuyển đổi 1 lần 10 trang pdf với tài khoản bình thường.

c) *Hãy mô tả cách thức để có thể xử lý một request với thời gian lớn mà không bị “request time out”. Hãy trích tối đa 10 dòng code trong bài tập đã làm thể hiện điều đó! (có thể thêm comment nếu cần, nhưng không được trích quá 10 dòng code)*

 # Trả lời: viết câu trả lời vào bên dưới

Tạo luồng riêng để xử lý tác vụ có thời gian chờ lớn. Khi tác vụ xử lý xong thì cập nhật trạng thái tác vụ lên cơ sở dữ liệu. Khi client muốn xem trạng thái tác vụ thì chỉ cần gọi câu lệnh để Server truy cập vào cơ sở dữ liệu để lấy trạng thái ra.

```
for (Part pdfPart : pdfParts) {
        if (pdfPart.getSubmittedFileName(); != null && ! pdfPart.getSubmittedFileName();.isEmpty()) {
            File uploadedFile = new File(outputPath + "/" + pdfPart.getSubmittedFileName(););
            pdfPart.write(uploadedFile.getAbsolutePath());
            pdfQueue.offer(uploadedFile);
        }
    }
    if (!isProcessing) {
        isProcessing = true;
        new Thread(new PdfProcessor(outputPath, userId)).start();
    }
```

Đà Nẵng, ngày 17 tháng 12 năm 2024