**BÀI KIỂM TRA SỐ 1**

1. Họ tên SV: Lê Trọng Nhân

2. Mã số SV: 20173292

3. Đã hoàn thành chương trình WebServer hỗ trợ 1 client: Đã hoàn thành

4. Phân tích code và so sánh với giao thức HTTP để giải thích các hoạt động của WebServer khi nhận yêu cầu và phục vụ client:

- Về cơ bản, đoạn code giúp hỗ trợ xử lý request qua HTTP Get nên hầu hết đều sẽ giống với giao thức HTTP, tuy nhiên do chỉ xử lý ở dòng GET nên các thông điệp phía sau GET (cùng nằm trong HTTP request) không được xử lý.

- Đầu tiên, server sẽ nhận vào toàn bộ request dưới dạng Input Stream. Sau đó, code tách request theo từng khoảng trắng, tìm đến vị trí đầu dòng GET, sau GET sẽ là /<tên file>. Có được tên file, chương trình sẽ tiến hành truy cập file và chuyển vài FileInputStream để in ra màn hình theo dạng HTTP Response với phần phiên bản, mã trả về, phần Content Type, Content Length và và body là nội dung của file.

5. Tạo một trang Web có form gồm trường text và button Download. Người sử dụng đưa tên file ảnh vào trường Text và click Download để kết nối WebServer và download ảnh. Điều gì xảy ra khi thiết lập kiểu submit là POST?

Khi thiết lập kiểu submit là POST thì tên của file submit từ form sẽ được lưu vào phần HTTP Body, ngoài ra HTTP Request cũng thay trường GET bằng trường POST.

6. Bổ sung code để WebServer có thể hỗ trợ download ảnh với POST. Copy phần code xử lý POST và giải thích

String action = tokenizedLine.nextToken();

if (action.startsWith("/")) action = action.substring(1);

System.out.println("This request is trying to " + action + " file");

String s = null;

while((s = input.readLine()).length() != 0) System.out.println(s);

StringBuilder payload = new StringBuilder();

while (input.ready()) payload.append((char) input.read());

String fileName = payload.toString().substring(5);

System.out.println(fileName);

try {

File file = new File(fileName);

int numOfBytes = (int) file.length();

FileInputStream inFile = new FileInputStream(fileName);

byte[] fileInBytes = new byte[numOfBytes];

inFile.read(fileInBytes);

output.writeBytes("HTTP/1.0 200 \r\n");

output.writeBytes("Content-Disposition: attachment; filename=" + fileName + "\r\n");

if (fileName.endsWith(".jpg"))

output.writeBytes("Content-Type: image/jpeg\r\n");

if (fileName.endsWith(".gif"))

output.writeBytes("Content-Type: image/gif\r\n");

output.writeBytes("Content-Length: " + numOfBytes + "\r\n");

output.writeBytes("\r\n");

output.write(fileInBytes, 0, numOfBytes);

} catch (Exception e) {

output.writeBytes("HTTP/1.0 404 File not found\r\n");

output.writeBytes("\r\n");

output.writeBytes("Excepton in reading file: " + e.getMessage());

}

Ta kiểm tra lần lượt phần header xem có phải là giao thức Post không, nếu đúng thì ta lấy tên file trong phần body và gửi trả về cho client.

7. Khi WebServer đang bận phục vụ 1 client download 1 file ảnh lớn, 1 client khác yêu cầu kết nối và truy nhập đến một ảnh nhỏ, điều gì xảy ra? Phân tích code & giải thích câu trả lời

Hệ thống sẽ phải chờ đợi file lớn download xong mới thực hiện download file nhỏ. Điều này là do chương trình là tuần tự, phải thực hiện đọc file và gửi trả phản hồi cho client đầu trước khi nhận thông điệp từ client sau để trả lời.

8. Bổ sung xử lý đa luồng cho WebServer để cùng lúc hỗ trợ nhiều client song song (copy phần code xử lý đa luồng và giải thích)

Tạo một class mới extends Thread để xử lý đa luồng, truyền vào tham số là connectionSocket để xử lý trên từng luồng.

Trên class mới:

private Socket connectionSocket;

public MultiThreadServerHandle(Socket s) {

this.connectionSocket = s;

}

Trên class cũ:

ServerSocket listenSocket = new ServerSocket(tcp\_port); // open listen port

System.out.println("Web server is listening at port " + tcp\_port);

while(true) {

Socket connectionSocket = listenSocket.accept();

MultiThreadServerHandle thread = new MultiThreadServerHandle(connectionSocket);

thread.start(); }