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:…Võ Minh Huy …**Lớp**:…21TCLC_DT1 ……**MSSV**:……102210061…...

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

a) Server:
   a. Lắng nghe và chấp nhận kết nối từ các client. Ứng với mỗi kết nối tạo 1 luồng xử lý sau:
      i. Chọn ngẫu nhiên 1 chuỗi **x** đúng 30 các ký tự alphabet, ví dụ: **x**="PhamMinhTuanPhamMinhTuanPhamMi"
      ii. Nhận 1 chuỗi từ client gửi tới.
      iii. Nếu chuỗi **y** từ client gửi tới **chứa các ký tự khác alphabet** thì thì trả về cho client chuỗi "**Khong dung dinh dang**".
      iv. Nếu chuỗi **y** đúng định dạng thì so sánh chuỗi đó với **x**.
      - Nếu **y** lớn hơn **x** theo thứ tự từ điển thì trả về cho client ký tự ">", nếu nhỏ hơn thì trả về ký tự "<" và tiến tới bước (v). Ví dụ: **y** = "xxx" thì phản hồi ">", **y** = "aaaaaa" thì phản hồi "<"
      - Nếu **y** bằng **x** theo thứ tự từ điển thì trả về "**Chuc mung vi da doan dung**" và nhảy tới bước (vi).
      v. Quay về bước (ii).
      vi. Đóng kết nối.

b) Client:
   a. Kết nối tới server
   b. Lặp đi lặp lại các bước sau:
      i. Nhập từ bàn phím một chuỗi alphabet. Nếu được thì có thể gợi ý nên nhập chuỗi gì!!
      ii. Gửi cho server số đã nhập
      iii. Nhận chuỗi ký tự từ server gửi về. Nếu chuỗi đó là "**Chuc mung vi da doan dung"** thì thoát khỏi vòng lặp
   c. Đóng kết nối

Yêu cầu server phải phản hồi request của client trong vòng không quá 1 giây!!! Nếu client có gợi ý thì mới được điểm tối đa.

 # Trả lời:
Dán code server vào bên dưới

```
package ThiCK;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
```

```
import java.net.Socket;
import java.time.DateTimeException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Random;

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

	public ChuoiServer() {
		try (ServerSocket server = new ServerSocket(3456)) {
			while (true) {
				Socket soc = server.accept();
				Xuly x = new Xuly(soc, this);
				x.start();
			}
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}

class Xuly extends Thread {
	ChuoiServer server;
	Socket soc;

	public Xuly(Socket soc, ChuoiServer server) {
		this.server = server;
		this.soc = soc;
	}

	public static boolean isAlphabet(String y) {
		String regex = "^[a-zA-Z]+$";
		return y.matches(regex);
	}

	private String randomString(int length) {
	  String characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
	  StringBuilder randomString = new StringBuilder();

	  Random random = new Random();
	  for (int i = 0; i < length; i++) {
	      int index = random.nextInt(characters.length());
	      randomString.append(characters.charAt(index));
	  }

	  return randomString.toString();
    }

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

			String x = randomString(30);
			while (true) {
				String clientInput = dis.readUTF().trim();
				System.out.println("From client: " + clientInput);

				if (isAlphabet(clientInput)) {
					int comparison = clientInput.compareTo(x);

					if (comparison > 0) {
```

```
                                    dos.writeUTF(">");

                              } else if (comparison < 0) {
                                    dos.writeUTF("<");
                              } else {
                                    dos.writeUTF("Chuc mung vi da doan dung");
                              }
                        } else {
                              dos.writeUTF("Khong dung dinh dang");
                        }
                  }
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
}
```

Dán code client vào bên dưới

```
package ThiCK;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;


public class ChuoiClient {
      public static void main(String[] args) {
            new ChuoiClient();

      }

      public ChuoiClient() {
            try {
                  Socket soc = new Socket("localhost", 3456);
                  DataInputStream dis = new DataInputStream(soc.getInputStream());
                  DataOutputStream dos = new
DataOutputStream(soc.getOutputStream());

                  BufferedReader inFormUser = new BufferedReader(new
InputStreamReader(System.in));

                  while (true) {
                        System.out.println("Nhap vao mot chuoi alphabet, or 'exit'
de ket thuc: ");
                        String txt = inFormUser.readLine();

                        if ("exit".equalsIgnoreCase(txt)) {
                              break;
                        }
                        dos.writeUTF(txt);
                        String message = dis.readUTF();
                        System.out.println("From server: " + message);

                  }

                  dis.close();
                  dos.close();
                  soc.close();
            } catch (Exception e) {
                   e.printStackTrace();
```

```
            }
        }
}
```

Dán các kết quả thực thi vào bên dưới

```
Nhap vao mot chuoi alphabet, or 'exit' de ket thuc:
#adf
From server: Khong dung dinh dang
Nhap vao mot chuoi alphabet, or 'exit' de ket thuc:
aaa
From server: >
Nhap vao mot chuoi alphabet, or 'exit' de ket thuc:
xxx
From server: >
Nhap vao mot chuoi alphabet, or 'exit' de ket thuc:
JRUvjjpkrmBReLMOHOCELQJmgrrvmN
From server: Chuc mung vi da doan dung
Nhap vao mot chuoi alphabet, or 'exit' de ket thuc:
```

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

 # Trả lời:

1. Thay thế Socket và ServerSocket bằng DatagramSocket và DatagramPacket.

2. Thay vì sử dụng DataInputStream và DataOutputStream, sử dụng DatagramPacket để nhận và gửi dữ liệu.

3. Thay vì sử dụng readUTF() và writeUTF(), bạn sẽ sử dụng DatagramPacket để nhận và gửi mảng byte.

- Tạo một mảng byte để chứa dữ liệu,
- Tạo một DatagramPacket từ mảng byte trên
- Sao đó dùng DatagramSocket: send('packet') để gửi cho server, receive('packet') để nhận từ server

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
Võ Minh Huy

Nguyễn Tấn Lộc
Võ Văn Đạt

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

Upload+ Convert : "/UploadFile"

1. Nhận request từ form upload file
2. Kiểm tra request có file không
3. Lấy ra User vừa gửi request
4. Tạo đối tượng UploadFileBO để thực hiện upload file pdf lên lưu trữ trên server, trong UploadFileBO:
   - Lặp qua các file
   - Lấy tên file -> format tên lại
   - Upload file vào thư mục "../webapp/pdfs" của server -> nếu lỗi: tạo 1 bảng ghi trong DB với userid, tên file, và status = 1 (nghĩa là "upload file error"); nếu thành công thì status = 0 (nghĩa "processing");
5. Khi upfile xong, thêm request vào hàng đợi, đẩy ra một luồng khác để xử lý. Sau đó request sẽ được lấy ra từ hàng đợi, và được gọi -> tạo ConvertFileBO:
   - Truy vấn lấy ra danh sách file có status = 0 ("processing") từ database với userid của người dùng
   - Lặp qua danh sách, từ fname lấy ra file pdf đã lưu trên server -> tiến hành convert và lưu vào thư mục "../webapp/docs". Nếu convert lỗi,trên db thay đổi stutus = 3 ("convert error) , nếu thành công thì status = 2 ("success").
   - Trả về message "Convert Completed", chuyển sang trang profile.

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

- Dùng Queue cho các yêu cầu: Tạo một hàng đợi (requestQueue) để giữ các yêu cầu HTTP được gửi từ phía client. Thay vì xử lý convert ngay lập tức, yêu cầu được thêm vào hàng đợi và đẩy sang luồng khác để xử lý. Sau đó gọi các yêu cầu và xử lý trong 1 luồng đồng bộ.

```
private final Queue<HttpServletRequest> requestQueue = new
ConcurrentLinkedQueue<>();
      protected void doGet(…){
        try {if (request.getPart("files").getSize() != 0) {
            // ... xử lý upload file lên server , lưu thông tin upload(userid,
fname, status) lên database
            requestQueue.offer(request); // đẩy request vào queue
            // Tạo một luồng khác để xử lý convert
            Thread newThread = new Thread(() -> {
                  processNextRequest(user);
            });
            newThread.start();
            // ... trả về thông báo, chuyển hướng}
        }} catch (Exception e) {}} // Bắt lỗi

      private void processNextRequest(UserBean user) {
          HttpServletRequest nextRequest = requestQueue.poll();  //lấy request từ
Queue
            if (nextRequest != null) { synchronized (this) {
                  new ConvertFileBO(user).run();}}}
```

Đà Nẵng, ngày 12 tháng 12 năm 2023