

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN LÍ THUYẾT GIỮA KỲ**

**MẠNG MÁY TÍNH**



**ĐỒ ÁN: REMOTE CONTROL**

**Giảng viên lý thuyết: Thầy Đỗ Hoàng Cường**

**Lớp: 21CTT2**

**Nhóm thực hiện: 14**

*Thành Phố Hồ Chí Minh, tháng 12 năm 2022*

# **MỤC LỤC**

[**MỤC LỤC** 2](#_Toc120699217)

[**A.** **Thông tin đồ án** 3](#_Toc120699218)

[I. Thông tin thành viên 3](#_Toc120699219)

[II. Phân chia công việc và mức độ hoàn thành 3](#_Toc120699220)

[III. Ý tưởng chính của đồ án 3](#_Toc120699221)

[IV. Môi trường lập trình và các thư viện ngoài hệ thống 3](#_Toc120699222)

[**B.** **Mô tả chức năng của các hàm, thủ tục (chia làm 2 project: Client và Server)** 3](#_Toc120699223)

[I. Project: Client 3](#_Toc120699224)

[1. Application (*Application.java*) 4](#_Toc120699225)

[2. ClientFrame (*ClientFrame.java*) 5](#_Toc120699226)

[3. ClientHandler (*ClientHandler.java*) 7](#_Toc120699227)

[4. KILL (*KILL.java*) 9](#_Toc120699228)

[5. Keylogger (*Keylogger.java*) 10](#_Toc120699229)

[6. Process (*Process.java*) 12](#_Toc120699230)

[7. START (*START.java*) 13](#_Toc120699231)

[8. Screen (*Screen.java*) 13](#_Toc120699232)

[II. Project: Server 15](#_Toc120699233)

[1. ServerFrame (*ServerFrame.java*) 15](#_Toc120699234)

[2. ServerHandler (*ServerHandler.java*) 16](#_Toc120699235)

# **Thông tin đồ án**

1. Thông tin thành viên

Bảng thông tin thành viên trong đồ án:

|  |  |
| --- | --- |
| **MÃ SỐ SINH VIÊN** | **HỌ VÀ TÊN SINH VIÊN** |
| 21120058 | Phạm Nhật Duy |
| 21120102 | Nguyễn Trúc Nguyên |
| 21120146 | Lê Nguyễn Phương Thùy |
| 21120150 | Nguyễn Song Toàn |
| 21120158 | Trương Công Trung |
| 21120179 | Nguyễn Đặng Đăng Khoa |

1. Phân chia công việc và mức độ hoàn thành

Bảng thông tin thành viên trong đồ án:

|  |  |  |
| --- | --- | --- |
| **CÔNG VIỆC** | **NGƯỜI THỰC HIỆN** | **MỨC ĐỘ HOÀN THÀNH** |
| Giao diện | 21120150 | 100% |
| Application | 21120150 | 100% |
| Process | 21120102 | 100% |
| Screenshot | 21120146 | 100% |
| Keylogger | 21120058 + 21120158 | 100% |
| Shutdown | 21120179 | 100% |
| Quay video | 21120058 + 21120179 | 100% |

1. Ý tưởng chính của đồ án

* Xác định kiến trúc mạng: Client – Server
* Giao thức sử dụng tầng Transport: TCP (Transmission Control Protocol)

Giao thức TCP cung cấp các dịch vụ truyền thông cho các ứng dụng mạng: truyền dữ liệu tin cậy. TCP cho phép các ứng dụng trên các máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo dữ liệu gửi từ bên gửi đến bên nhận chính xác và đúng thứ tự.

* Port được sử dụng: 8888
* Vì số kết nối giữa client và server là 1 kết nối, nên khi thực hiện những chức năng mà có hiển thị ra form để thực hiện, thì khi thực hiện xong, sẽ tắt form đó để thực hiện tiếp những chức năng khác. Nếu không tắt form, sẽ dẫn đến lỗi lặp.

1. Môi trường lập trình và các thư viện ngoài hệ thống

* Phần mềm biên soạn và biên dịch ngôn ngữ lập trình Java: Netbeans
* Thư viện hỗ trợ cho chức năng Keylogger: *jnativehook-2.1.0.jar*

# **Mô tả chức năng của các hàm, thủ tục** **(chia làm 2 project: Client và Server)**

1. Project: Client

*Package: client gồm các class:*

* + - 1. Application (*Application.java*)

*Chức năng:* Hiển thị form gồm các nút bấm để xử lí, gửi tín hiệu đến server để thực hiện các chức năng liên quan đến application: list running apps, kill app, start app và xóa các dòng trong bảng list running apps. Gửi request qua server để thực hiện các chức năng và nhận response từ server.

Graphical user interface, application

Description automatically generated

***Các hàm, thủ tục:***

+ Khai báo các biến cSocket, dOut, dIn và tạo Constructor Application với tham chiếu cSocket:

Text

Description automatically generated

+ Hàm *sendSignal(String s, Socket sc)* với tham chiếu s, sc dùng để ghi dữ liệu để gửi request cho server thông qua biến *dOut*.

+ Hàm *jButton1ActionPerformed()* được thực hiện khi nhấn nút LIST, lúc này sẽ gửi tín hiệu “LISTAPP” qua server thông qua hàm sendSignal(“LISTAPP”, cSocket), nhận số lượng dòng kết quả khi chạy lệnh cmd từ server lưu vào biến apps, sau đó xử lí từng dòng và tách chuỗi để được 3 biến: name, id, countThread. Sau đó, ghi từng dòng với 3 cột: name, id, countThread vào Table để hiển thị ra cửa sổ Application.

+ Hàm *jButton2ActionPerformed()* được thực hiện khi nhấn nút KILL, lúc này sẽ gửi tín hiệu “KILLAPP” qua server thông qua hàm sendSignal(“KILLAPP”, cSocket) và hiển thị form của class KILL.

+ Hàm *jButton3ActionPerformed()* được thực hiện khi nhấn nút DELETE, lúc này xóa hết các dòng của Table ở cửa sổ Application.

+ Hàm *jButton4ActionPerformed()* được thực hiện khi nhấn nút START, lúc này sẽ gửi tín hiệu “STARTAPP” qua server thông qua hàm sendSignal(“STARTAPP”, cSocket) và hiển thị form của class START.

* + - 1. ClientFrame (*ClientFrame.java*)

***Chức năng:*** Hiển thị form kết nối đến server và xử lí vấn đề kết nối đến server.

Graphical user interface, application

Description automatically generated

***Các hàm, thủ tục:***

+ Khai báo các biến để sử dụng cho toàn bộ class

A picture containing graphical user interface

Description automatically generated

+ Hàm initComponents() là hàm do hệ thống Netbeans tạo ra khi người lập trình thực hiện tạo form trên phần Design.

Text

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

+ Hàm jButton1ActionPerformed() được thực hiện khi nhấn vào nút bấm Connect ở Login Client form. Hàm này để kiểm tra IP và port có đúng không, nếu đúng thì thực hiện kết nối với server và gọi class ClientHandler, hiển thị form của class ClientHandler và hiển thị sẽ hiển thị MessageDialog Connected successfully.

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

* + - 1. ClientHandler (*ClientHandler.java*)

Chức năng: Hiển thị form gồm các nút bấm thực hiện các chức năng lớn: APP, PROCESS, SHUTDOWN, KEYSTROKE (KEYLOGGER), SCREENSHOT và thoát chương trình. Gửi request qua server và nhận response từ server.

Diagram

Description automatically generated

Các hàm, thủ tục:

+ Khai báo các biến cần dùng cho class: cSocket (Kiểu Socket), DataInputStream *dIn* (dùng để đọc dữ liệu từ luồng bên ngoài vào), DataOutputStream *dOut* (để ghi dữ liệu mà sau này có thể được đọc bởi một DataInputStream)

+ Tạo constructor để sử dụng qua các class khác với dữ liệu đầu vào là Socket

Graphical user interface, text, application

Description automatically generated

+ Hàm *sendSignal(String s, Socket sc)* với tham chiếu s, sc dùng để ghi dữ liệu để gửi request cho server thông qua biến *dOut*.

+ Hàm *jButton1ActionPerformed()* được thực hiện khi nhấn nút APPLICATION, lúc này sẽ gửi tín hiệu “APP” qua server thông qua hàm sendSignal(“APP”, cSocket) và hiển thị form của class Application.

+ Hàm *jButton2ActionPerformed()* được thực hiện khi nhấn nút PROCESS, lúc này sẽ gửi tín hiệu “PROCESS” qua server thông qua hàm sendSignal(“PROCESS”, cSocket) và hiển thị form của class Process.

+ Hàm *jButton3ActionPerformed()* được thực hiện khi nhấn nút KEYSTROKE, lúc này sẽ gửi tín hiệu “KEYLOGGER” qua server thông qua hàm sendSignal(“KEYLOGGER”, cSocket) và hiển thị form của class Keylogger.

+ Hàm *jButton4ActionPerformed()* được thực hiện khi nhấn nút SCREENSHOT, lúc này sẽ gửi tín hiệu “SCREEN” qua server thông qua hàm sendSignal(“SCREEN”, cSocket) và hiển thị form của class Screen.

+ Hàm *jButton5ActionPerformed()* được thực hiện khi nhấn nút EXIT, lúc này sẽ gửi tín hiệu “EXIT” qua server thông qua hàm sendSignal(“EXIT”, cSocket) và đóng kết nối của client, tắt cửa sổ chương trình

+ Hàm *jButton6ActionPerformed()* được thực hiện khi nhấn nút SHUTDOWN, lúc này sẽ gửi tín hiệu “SHUTDOWN” qua server thông qua hàm sendSignal(“SHUTDOWN”, cSocket).

+ Hàm *formWindowClosed()* được thực hiện khi nhận nút X của cửa sổ chương trình. Hàm này sẽ sẽ gửi tín hiệu “EXIT” qua server thông qua hàm sendSignal(“EXIT”, cSocket) và đóng kết nối của client, tắt cửa sổ chương trình.

* + - 1. KILL (*KILL.java*)

Chức năng: Hiển thị form để điền ID của app hoặc process cần kill và thực hiện gửi request đến server để Kill app hoặc process bằng ID, nhận response từ server.

Các hàm, thủ tục:

+ Khai báo các biến cSocket, dIn, dOut và tạo constructor KILL

Graphical user interface, text, application

Description automatically generated

+ Hàm *jButton1ActionPerformed()* được thực hiện khi nhấn nút KILL, lấy ID từ textField của người dùng nhập vào lưu vào biến *string s* ở Form KILL, gửi string s đến server qua *dOut*, server xử lý và sau đó nhận dữ liệu từ server gửi về qau *dIn*, nếu nhận byte 1 thì hiển thị MessageDialog “Kill successfully!”, ngược lại thì hiển thị MessageDialog “Kill unsuccessfully!”.

+ Hàm *formWindowClosed()* được thực hiện khi nhấn nút đóng cửa sổ KILL. Hàm này gửi tín hiệu “EXITKILL” đến server qua dOut và đóng cửa sổ form KILL.

* + - 1. Keylogger (*Keylogger.java*)

**Chức năng:**

Hiển thị form gồm các nút bấm để xử lí, gửi tín hiệu đến server để thực hiện các chức năng liên quan đến Key logger : bắt đầu bắt phím của máy Server (TAKE), dừng việc bắt phím ở máy server, in các phím đã bắt được ra màn hình và xóa màn hình . Gửi request qua server để thực hiện các chức năng và nhận response từ server.

Graphical user interface, application

Description automatically generated

**Các hàm, thủ tục:**

+ Khai báo các biến cSocket, dOut, dIn, img và tạo Constructor Keylogger với tham chiếu cSocket:

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Hàm sendSignal(String s, Socket sc) với tham chiếu s, sc dùng để ghi dữ liệu để gửi request cho server thông qua biến dOut.

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Hàm jButton1ActionPerformed() được thực hiện khi nhấn nút HOOK. Lúc này sẽ gửi tín hiệu “HOOK” qua server thông qua hàm sendSignal(“HOOK”, cSocket). Khi nhận request từ client, server sẽ tiến hành bắt phím nhấn và lưu vào một mảng được khởi tạo trước. Vì thế có thể bắt được 1 lần rất nhiều phím.

Ảnh có chứa văn bản

Mô tả được tạo tự động

A picture containing diagram

Description automatically generated

ServerHandler.java

+ Hàm jButton2ActionPerformed() được thực hiện khi nhấn nút UNHOOK. Lúc này sẽ gửi tín hiệu “UNHOOK” qua server thông qua hàm sendSignal(“UNHOOK”,cSocket). Khi nhận request từ client, server sẽ dừng việc bắt phím nhấn lại.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Text

Description automatically generated

+ Hàm jButton3ActionPerformed() được thực hiện khi nhấn nút PRINT. Lúc này sẽ gửi tín hiệu “PRINT” qua server thông qua hàm sendSignal(“PRINT”,cSocket). Server nhận request “PRINT” từ client thông qua và sẽ trả về client mảng các phím đã bắt được rồi in ra màn hình

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Hàm jButton4ActionPerformed() được thực hiện khi nhấn nút DELETE. Lúc này sẽ gửi tín hiệu “DELETE” qua server thông qua hàm sendSignal(“DELETE”,cSocket). Khi đó mảng các nút bấm bắt được được in ra trên màn hình sẽ bị xóa.

* + - 1. Process (*Process.java*)

***Chức năng:*** Hiển thị form gồm các nút bấm để xử lí, gửi tín hiệu đến server để thực hiện các chức năng liên quan đến process: list process, kill process, start process và xóa các dòng trong bảng list process. Gửi request qua server để thực hiện các chức năng và nhận response từ server.

*Các hàm, thủ tục:*

+ Khai báo các biến cSocket, dOut, dIn và tạo Constructor Process với tham chiếu cSocket:

Graphical user interface, text, application, email

Description automatically generated

+ Hàm *sendSignal(String s, Socket sc)* với tham chiếu s, sc dùng để ghi dữ liệu để gửi request cho server thông qua biến *dOut*.

Text

Description automatically generated with medium confidence

+ Hàm *jButton1ActionPerformed()* được thực hiện khi nhấn nút LIST, lúc này sẽ gửi tín hiệu “LISTPRO” qua server thông qua hàm sendSignal(“LISTPRO”, cSocket), nhận số lượng dòng kết quả khi chạy lệnh cmd từ server lưu vào biến pros, sau đó xử lí từng dòng và tách chuỗi để được 5 biến: name, id, sessionName, session#. Sau đó, ghi từng dòng với 5 cột trên vào Table để hiển thị ra cửa sổ Process.

+ Hàm *jButton2ActionPerformed()* được thực hiện khi nhấn nút KILL, lúc này sẽ gửi tín hiệu “KILLPRO” qua server thông qua hàm sendSignal(“KILLPRO”, cSocket) và hiển thị form của class KILL.

+Hàm *jButton3ActionPerformed()* được thực hiện khi nhấn nút DELETE, lúc này xóa hết các dòng của Table ở cửa sổ Process

+ Hàm *jButton4ActionPerformed()* được thực hiện khi nhấn nút START, lúc này sẽ gửi tín hiệu “STARTAPP” qua server thông qua hàm sendSignal(“STARTAPP”, cSocket) và hiển thị form của class START.

* + - 1. START (*START.java*)

**Chức năng:** Hiển thị form để xử lí, gửi tên app cần bật đến server và nhận tín hiệu thành công hay không từ server.

* + - 1. Screen (*Screen.java*)

***Chức năng***

Hiển thị form gồm các nút bấm để xử lí, gửi tín hiệu đến server để thực hiện các chức năng liên quan đến ScreenShot: Chụp ảnh màn hình của máy Server (TAKE) và lưu ảnh chụp màn hình vào vị trí tùy chọn trong máy Client (SAVE). Gửi request qua server để thực hiện các chức năng và nhận response từ server.

Graphical user interface, application, Word

Description automatically generated

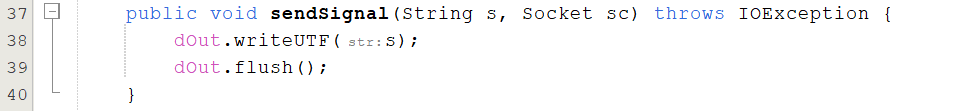
***Các hàm, thủ tục***

+ Khai báo các biến cSocket, dOut, dIn, img và tạo Constructor Screen với tham chiếu cSocket:

***Text

Description automatically generated***

+ Hàm sendSignal(String s, Socket sc) với tham chiếu s, sc dùng để ghi dữ liệu để gửi request cho server thông qua biến dOut.



+ Hàm jButton1ActionPerformed() được thực hiện khi nhấn nút TAKE, lúc này sẽ gửi tín hiệu “SCREENSHOT” qua server thông qua hàm sendSignal(“SCREENSHOT”, cSocket), server nhận request “SCREENSHOT” từ client thông qua và trả về response thông qua DataOutputStream. Khi nhận request từ client, server sẽ tiến hành chụp màn hình và lưu vào bộ nhớ tạm dưới dạng mảng. Client sau đó tiến thành tạo một biến có kích thước phù hợp và đọc mảng được lưu tạm này vào biến với kích thước vừa đủ. Từ biến đó chuyển thành hình ảnh dạng icon và hiển thị lên cửa sổ của chức năng Screenshot để người dùng ở client có thể xem ảnh vừa chụp trước khi lưu. Do đó người dùng có thể chụp rất nhiều ảnh chụp màn hình cho đến khi quyết định lưu.

Graphical user interface, text, application

Description automatically generated

+ Hàm jButton2ActionPerformed() được thực hiện khi nhấn nút SAVE, lúc này trong Client sử dụng chức năng File Choose để hiển thị hộp chọn địa chỉ lưu. Sau đó người dùng sẽ chọn vị trí muốn lưu ảnh trong hộp và nhập tên của ảnh theo ý muốn, kèm đuôi có định dạng phù hợp để lưu ảnh thành công.

Text, application

Description automatically generated

1. Project: Server

Package: server

ServerFrame (*ServerFrame.java*)

Chức năng: Hiển thị form để hiển thị địa chỉ IP và port 8888 (port cố định) của máy server. Gọi class ServerHandler.

Graphical user interface, text, application

Description automatically generated

ServerHandler (*ServerHandler.java*)

***Chức năng:***

1. Khởi tạo ServerSocket với port 8888 (port cố định)
2. Accepct kết nối từ client để thực hiện nhận request từ client và trả về reponse.
3. Xử lí, thực hiện các lệnh liên quan đến chức năng khi nhận được request thực hiện chức năng đó từ client.
4. Trả về kết quả thực hiện chức năng về cho client.

***Các lệnh thực hiện các chức năng:***

+ Khai báo biến clients để đánh dấu để kết thúc vòng lặp bên dưới, nghĩa là khi nhận tín hiệu “EXIT” từ client, biến clients sẽ tăng lên 1 để thoát vòng lặp. Biến serverC được khởi tạo, biến client dùng để chấp nhận kết nối từ phía client nếu đúng IP, Port.

+ Trong vòng while (clients == 0) sẽ đọc các request từ client (khi phía client bấm vào các nút ở form của ClientHandler) gửi qua server và lưu vào biến *re*.

+ Phần lệnh if (re.equals(“APP”)), nếu nhận được request “APP” từ cửa sổ form ClientHandler ở phía client thì sẽ xử lý những chức năng: list running apps, kill running apps bằng ID, start app ở bên máy server khi nhận được những request ở phía client (“LISTAPP”, “KILLAPP”, “STARTAPP”). Bên trong lệnh if có 1 vòng lặp while (checkApp) dùng để lặp cho đến khi nhận được request “EXITAPP” từ client, tức là bên client đóng cửa sổ form Application. Đọc các request từ cửa sổ form Application của phía client và lưu vào biến *aRe*.

* Phần lệnh bên trong if (aRe.equals(“LISTAPP”)) sẽ thực hiện chức năng chạy process theo lệnh yêu cầu trong powershell bằng cách dùng biến Process, ProcessBuilder; đếm số dòng và đọc từng dòng kết quả trả về khi chạy lệnh trong powershell; gửi số lượng dòng, nội dung từng dòng về client thông qua dOut.
* Phần lệnh bên trong if (aRe.equals("KILLAPP")) sẽ thực hiện chức năng: nhận ID của app cần KILL bên client, xử lí chạy process theo lệnh cmd.exe để kill app bằng ID *“taskkill /PID {ID} /F”* bằng cách dùng các biến Process, ProcessBuilder và trả kết quả thực hiện bằng các byte 0, 1 (0: không thành công; 1: thành công).
* Phần lệnh bên trong if (aRe.equals("STARTAPP")) sẽ thực hiện chức năng: nhận tên app cần bật từ client và sau đó xử lí chạy process theo lệnh cmd.exe để bật app bằng tên: “start {name}.exe” bằng cách dùng các biến Process, ProcessBuilder và trả kết quả thực hiện bằng các byte 0, 1 (0: không thành công; 1: thành công).

+ Phần lệnh if (re.equals(“PRO”)), nếu nhận được request “PRO” từ cửa sổ form ClientHandler ở phía client thì sẽ xử lý những chức năng: list running process, kill running process bằng PID, start process ở bên máy server khi nhận được những request ở phía client (“LISTPRO”, “KILLPRO”, “STARTPRO”). Bên trong lệnh if có 1 vòng lặp while (checkPro) dùng để lặp cho đến khi nhận được request “EXITAPP” từ client, tức là bên client đóng cửa sổ form Process. Đọc các request từ cửa sổ form Process của phía client và lưu vào biến *aRe*.

* Phần lệnh bên trong if (aRe.equals(“LISTPRO”)) sẽ thực hiện chức năng chạy process theo lệnh yêu cầu trong cmd bằng cách dùng biến Process, ProcessBuilder; đếm số dòng và đọc từng dòng kết quả trả về khi chạy lệnh trong cmd; gửi số lượng dòng, nội dung từng dòng về client thông qua dOut.
* Phần lệnh bên trong if (aRe.equals("KILLPRO")) sẽ thực hiện chức năng: nhận ID của app cần KILL bên client, xử lí chạy process theo lệnh cmd.exe để kill process bằng ID *“taskkill /F /PID +{ID};* bằng cách dùng các biến Process, ProcessBuilder và trả kết quả thực hiện bằng các byte 0, 1 (0: không thành công; 1: thành công).
* Phần lệnh bên trong if (aRe.equals(“STARTAPP”)) sẽ thực hiện chức năng: nhận tên process cần bật từ client và sau đó xử lí chạy process theo lệnh cmd.exe để bật process bằng tên: “start {name}.exe” bằng cách dùng các biến Process, ProcessBuilder và trả kết quả thực hiện bằng các byte 0, 1 (0: không thành công; 1: thành công)

+ Phần lệnh if (re.equals(“SCREENSHOT”)), nếu nhận được request “SCREENSHOT” từ cửa sổ form ClientHandler ở phía client thì sẽ xử lý chức năng: Chụp màn hình và lưu ảnh vào Output Stream dưới dạng Buffer. Bên trong lệnh if có 1 vòng lặp while (checkApp) dùng để lặp cho đến khi nhận được request “EXITSCREEN” từ client, tức là bên client đóng cửa sổ form Screenshot. Đọc các request từ cửa sổ form Screenshot của phía client và lưu vào biến *aRe*.

+ Phần lệnh if (re.equals(“SHUTDOWN”)):

* Hàm runtime.getRuntime():

*Chức năng:* Trả về đối tượng thời gian chạy được liên kết với phần mềm Java hiện tại.

*Cách hoạt động:* Lấy thời gian chạy hiện tại được liên kết với tiến trình, chương trình đang chạy.

* Hàm runtime.exec(command):

*Chức năng:* Thực thi lệnh chuỗi đã chỉ định trong một quy trình riêng biệt. Phương thức này trả về một đối tượng tiến trình mới để quản lý tiến trình con.

*Cách hoạt động:* Cụ thể trong bài làm, hàm sẽ là **runtime.exec("shutdown -s -t 5")**, và liên kết thực thi trong Command Prompt, sử dụng Shutdown command để điều khiển tắt máy server. Trong Shutdown command có hỗ trợ các lệnh tắt máy “**-s**” và hẹn giờ thực thi lệnh “**-t xxx**” (xxx: từ 0 đến 315360000 giây). Bài làm cài đặt tắt máy sau 5 giây để server có thể hiện thông báo tắt máy vừa đủ thời gian đọc. Sau khi thực hiện, màn hình sẽ thông báo máy của bạn sẽ tắt trong vòng vài giây/ít hơn 1 phút “Windows will shutdown in less than a minute.” (ảnh minh họa bên dưới)

Graphical user interface, website

Description automatically generated

* *Lỗi* IOException*:* Nếu xảy ra Input/Output Error, chương trình sẽ in ra Exception cụ thể.