ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN SOCKET

Môn: Mạng Máy Tính

Giáo Viên Hướng Dẫn

Thầy ĐỖ HOÀNG CƯỜNG
Thầy NGUYỄN THANH QUÂN
Cô HUỲNH THỊ BẢO TRÂN

LÊ NGÔ SONG CÁT- BÙI NGỌC KIỀU NHI - ĐẶNG VĨNH TƯỜNG

MỤC LỤC

BÁO CÁO ĐỒ ÁN SOCKET

I. THÔNG TIN THÀNH VIÊN

MSSV	Họ và tên
21127495	Lê Ngô Song Cát
21127659	Bùi Ngọc Kiều Nhi
21127720	Đặng Vĩnh Tường

II. PHÂN CHIA CÔNG VIỆC VÀ MỨC ĐỘ HOÀN THÀNH

Chức năng	Người thực hiện	Mức độ hoàn thành
Kết nối	21127659	100%
Chụp màn hình	21127495	100%
Tắt máy	21127495	100%
Process Running	21127720	100%
App Running	21127720	100%
Keystroke	21127495	100%
Giao diện	21127659	100%
Ghép chương trình	21127720	100%
Chat	21127659	100%
Report	21127495	100%
	21127659	
	21127720	

III. MÔ TẢ CÁC HÀM/THỦ TỤC CỦA CHƯƠNG TRÌNH

1. Client

1.1. ChatMessageSocket.java

- Tạo Socket để gửi và nhận dữ liệu thông qua ObjectOutputStream và ObjectInputStream.
- Hàm send(): dùng để gửi một Object cho server.
- Hàm readData(): đọc một Object từ stream lên cho client xử lý.
- Hàm isActive(): cho biết hiện tại kết nối đã bị đóng chưa.
- Hàm close(): đóng kết nối.

1.2. UIManager.java

- Giao diện chính của chương trình phía client gồm các nút:
  + Connect: dùng để tạo kết nối với server ở Port 9000.
  + Send: gửi tin nhắn cho server.
- Sử dụng Consumer<String> của java để cài đặt chức năng cho các nút.
- Hàm addMessage(): hiện thị tin nhắn từ server gửi qua.
- Hàm addEvent(): thêm một chức năng cho giao diện.
- Hàm inform(): hiển thị một thông báo cho người dùng.
- Hàm UIManager() hiển thị giao diện phía client bao gồm các lựa chọn như: Connect, Send.

1.3. TaskManager.java

- Class TaskManager dùng để thực hiện các chức năng: tắt máy, tắt chương trình, liệt kê ứng dụng, liệt kê tiến trình
- Hàm findFullPath(): sử dụng chương trình where.exe của windows và chương trình reg.exe để tìm đường dẫn tuyệt đối đến file thực thi của ứng dụng.
- Hàm getTask(): dùng ProcessBuilder() để chạy chương trình tasklist.exe liệt kê các tiến trình đang chạy sau đó đọc và xử lý dữ liệu để lấy được tên, id, memory usage của ứng dụng.

- Hàm start(): sử dụng hàm findFullPath() để tìm ra đường dẫn tuyệt đối đến file thực thi của ứng dụng cần chạy, nếu tìm thấy thì hàm chạy ứng dụng và trả về True, nếu không tìm thấy thì trả về False.
- Hàm getAppRunning(): dùng ProcessBuilder() để chạy powershell và sử dụng lệnh Get-Process để liệt kê thông tin các ứng dụng đang chạy gồm tên, id, memory usage.
- Hàm shutdown(): thực hiện việc tắt máy bằng cách sử dụng ProcessBuilder() để chạy chương trình shutdown.exe của hệ điều hành windows.
- Hàm killProcess(): sử dụng class ProcessHandle() để tìm chương trình cần tắt bằng ID, nếu tìm thấy thì thực hiện diệt bằng hàm destroyForcibly và trả về True, nếu không tìm thấy thì trả về False.

1.4. KeyLogger.java

- Dùng thư viện JNativeHook để ghi lại các phím mà máy client nhấn.
- Hàm setLoggerOff(): tắt log của của GlobalScreen để chương trình không in ra các thông tin không liên quan.
- Hàm hook(): thêm NativeKeyListener để bắt đầu lắng nghe các phím mà client nhấn.
- Hàm unhook(): gỡ bỏ NativeKeyListener hiện có để dừng việc lắng nghe.
- Hàm convertKeyEvent(): chuyển thông tin nhận được khi bấm thành ký hiệu của các phím.
- Hàm nativeKeyPressed(): được gọi mỗi khi có một phím được nhấn, thêm các ký hiệu được trả về bởi hàm converKeyEvent vào buffer để có thể gửi cho server khi cần.
- Hàm getText(): trả về dữ liệu trong buffer và reset lại buffer.

1.5. Main.java

- Tạo một key logger để gửi đi các thông tin (các nút được nhấn) từ bàn phím.
- Tạo UIManager để hiện thị lên form giao diện.
- Hàm getScreenCapture() dùng để tạo ảnh chụp màn hình bằng hàm createScreenCapture() trong class Robot. Hàm trả về hình ảnh nhận được sau khi chụp màn hình.
- Hàm startListening(): chạy một luồng mới dành cho việc lắng nghe thông tin từ server. Nếu thông tin nhận được là một lệnh (String) thì thực hiện hàm executeStringCommand().
- Hàm excuteStringCommand(String) nhận vào một string đã được gửi từ server:
  + Nếu chuỗi nhận vào là "_SEND_SCREEN_CAPTURE_" thì sẽ gửi cho server kết quả từ hàm getScreenCapture() ở trên.
  + Nếu chuỗi nhận vào là "_SHUT_DOWN_" thì sẽ thực thi hàm shutdown() trong class TaskManager() nếu shutdown thành công gửi "$-Successful-$ ...", ngược lại gửi về server "$-Fail-$ ...".
  + Nếu chuỗi nhận vào là "_LIST_TASK_" thì thực thi hàm getTask() của TaskManager() và gửi dữ liệu qua cho server.
  + Nếu chuỗi nhận vào là "_LIST_APP_" thì thực thi hàm getAppRunning() của TaskManager() và gửi dữ liệu cho server.
  + Nếu chuỗi nhận vào là "_START_TASK_ + [tên ứng dụng]" thì thực hiện cắt chuỗi để lấy tên của chương trình cần bắt đầu và thêm vào đuôi ".exe" nếu chưa có rồi truyền vào hàm start() thuộc class TaskManager(). Nếu bắt đầu ứng dụng thành công gửi "$-Successful-$ ...", ngược lại gửi về server "$-Fail-$ ...".

+ Nếu chuỗi nhận vào là "_KILL_TASK_ + [id ứng dụng]" thì thực hiện cắt chuỗi để nhận vào id của ứng dụng cần tắt rồi truyền id vào hàm killProcess() của class TaskManager(). Nếu thực hiện tắt ứng dụng thành công gửi "$-Successful-$ ...", ngược lại gửi về server "$-Fail-$ ...".
+ Nếu chuỗi nhận vào là "_START_KEY_LOG_" thì bắt đầu ghi lại các phím được nhấn vào buffer của KeyLogger.
+ Nếu chuỗi nhận vào là "_STOP_KEY_LOG_" thì dừng việc ghi các phím được nhấn vào buffer của KeyLogger.
+ Nếu chuỗi nhận vào là "_SHOW_KEY_LOG_" thì lấy tất cả dữ liệu trong buffer của KeyLogger bằng hàm getText() và gửi cho server.
+ Nếu chuỗi nhận vào không bắt đầu bằng lệnh thì hiển thị tin nhắn từ server.

- Hàm addEvents(): cài đặt chức năng cho các nút trong giao diện của client bằng cách sử dụng lamda để gán giá trị cho các Consumer<String> của UIManager.

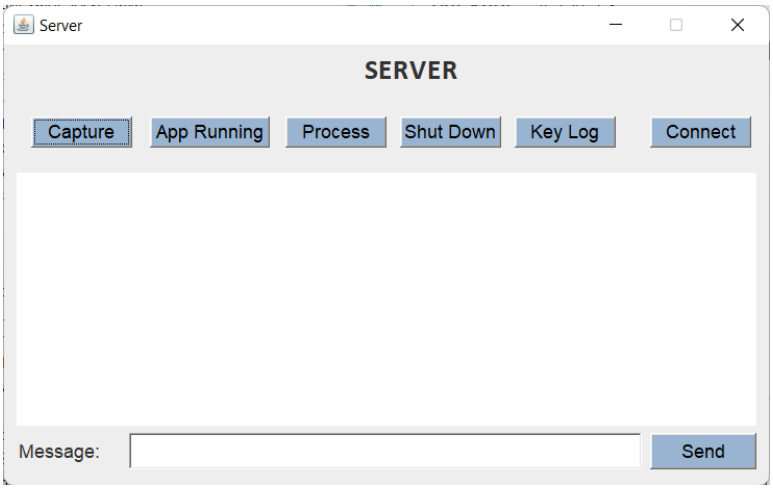
2. Server

2.1. ChatMessageSocket.java

- Tạo Socket để gửi và nhận dữ liệu thông qua ObjectOutputStream và ObjectInputStream.
- Hàm send(): dùng để gửi một Object cho client.
- Hàm readData(): đọc một Object từ stream lên cho server xử lý.
- Hàm isActive(): cho biết hiện tại kết nối đã bị đóng chưa.
- Hàm close(): đóng kết nối.

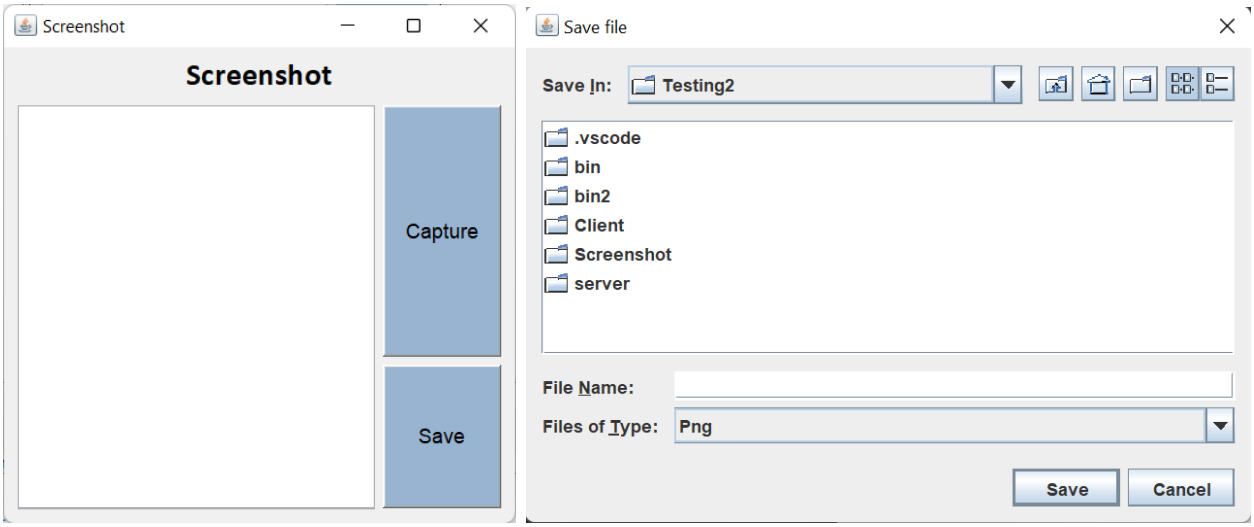
2.2. UIManager.java

- Giao diện chính của chương trình phía client gồm các nút:
  + Connect: dùng để tạo server ở Port 9000 và chờ client kết nối.
  + Send: gửi tin nhắn cho client.
  + Capture: dùng để mở giao diện "Screenshot" và thực hiện việc chụp ảnh màn hình thông qua nút "Capture" rồi lưu ảnh vừa chụp được bằng nút "Save".
  + App Running: dùng để mở giao diện "App Running" và thực hiện các thao tác như: liệt kê, "kill" các ứng dụng đang chạy, khởi động một ứng dụng mới trên máy client.
  + Process: dùng để mở giao diện "Process" và thực hiện các thao tác như: liệt kê, "kill" các tiến trình đang chạy, khởi động một tiến trình mới trên máy client.
  + Shut Down: dùng để gửi lệnh yêu cầu client tắt máy.
  + Key Log: dùng để mở giao diện "Keystroke" và thực hiện các thao tác như: Hook, Unhook, hiển thị các phím được gõ.
- Sử dụng Consumer<String> của java để cài đặt chức năng cho các nút.
- Hàm addMessage(): hiện thị tin nhắn từ client gửi qua.
- Hàm addEvent(): thêm một chức năng cho giao diện.
- Hàm inform(): hiển thị một thông báo cho người dùng.
- Hàm addKeyLogText(): đưa dữ liệu xuống cho KeystrokeMenu xử lý.
- Hàm addScreenShotImage(): đưa dữ liệu xuống cho ScreenShotMenu xử lý.
- Hàm displayProcess(): đưa dữ liệu xuống cho ProcessMenu xử lý.
- Hàm UIManager() hiển thị giao diện phía server bao gồm các lựa chọn như: Capture, App Running, Process, Shut Down, Key Log, Connect, Send.



2.3. ScreenshotMenu.java

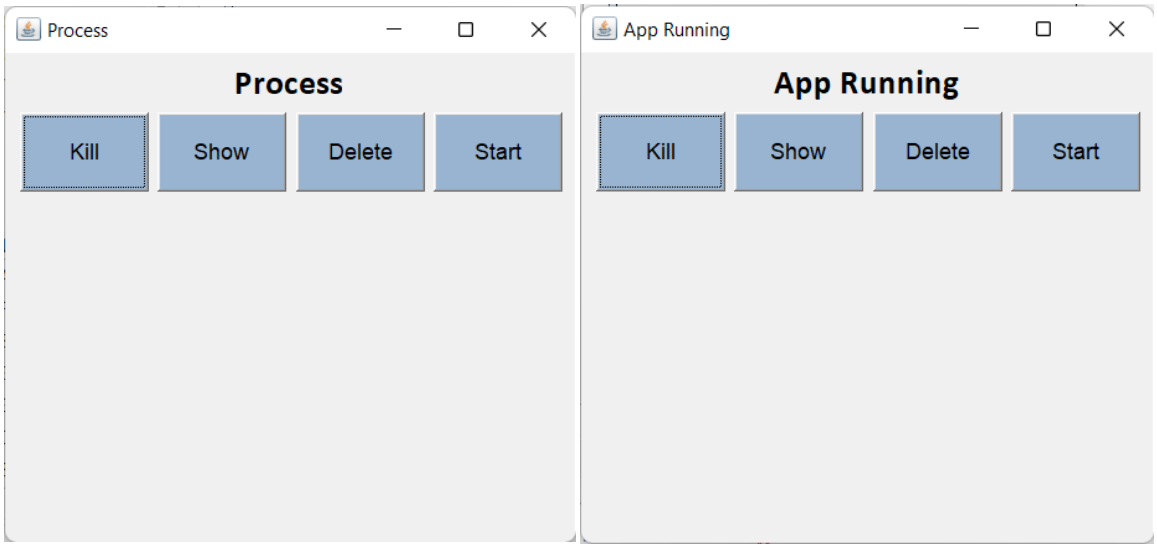
- Hàm addImage(): hiện thị ảnh nhận được.
- Hàm writeImageToFile(): dùng để ghi dữ liệu ảnh vào file bằng class ImageIO của java.
- Hàm ScreenshotMenu() hiển thị giao diện các lựa chọn cho server như : Capture, Save.
- Nút "Capture": gửi lệnh yêu cầu client gửi ảnh chụp màn hình và hiển thị ảnh.
- Nút "Save": hiện ra giao diện để người dùng chọn vị trí cũng như tên để lưu ảnh đã nhận được.



2.4. ProcessMenu.java

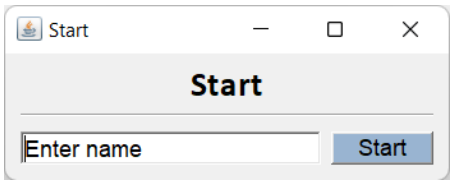
- Hàm displayTable(): hiển thị dữ liệu của các tiến trình hoặc ứng dụng đang chạy dưới dạng bảng.
- Hàm setTypeMenu(): dùng để chọn việc hiển thị cho Process hay App Running.
- Hàm setStartAction() dùng để gán vào hành động cần thực hiện là Start nếu server chọn hành động Start. Tương tự với các hàm setStopAction() và setShowAction().

- Hàm ProcessMenu() hiển thị giao diện các lựa chọn cho server như : Kill, Show, Delete, Start.
- Nút “Kill" sẽ gửi lệnh “_KILL_TASK_ + [ID]” qua cho client.
- Nút “Start” sẽ gửi lệnh “_START_TASK_ + [tên ứng dụng]” cho client.
- Nút “Show” sẽ gửi lệnh “_LIST_APP_” hoặc lệnh “_LIST_TASK_" cho client.

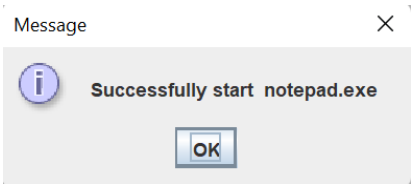


2.5. StartPAMenu.java

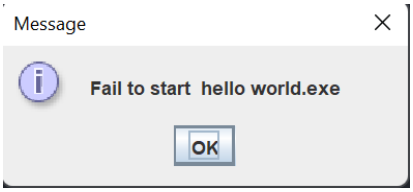
- Hàm StartPAMenu() hiển thị form để điền tên tiến trình cần khởi động. Nhập tên của tiến trình ta muốn khởi động và nhấn nút “Start”.



+ Nếu khởi động tiến trình thành công, giao diện sẽ hiện thị như sau:

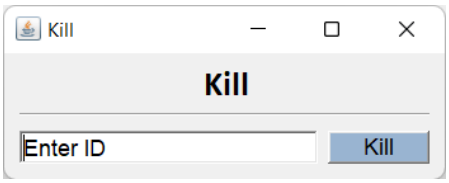


+ Nếu không tìm thấy tiến trình hoặc xảy ra lỗi khi khởi động tiến trình, giao diện sẽ hiển thị như sau:

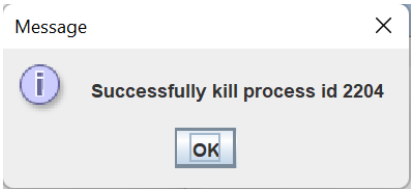


2.6. StopPAMenu.java

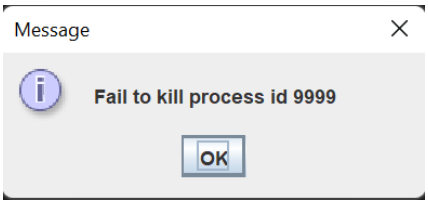
- Hàm StopPAMenu() hiển thị form để điền tên tiến trình muốn tắt đi. Nhập ID tên của tiến trình ta muốn tắt và nhấn nút "Kill".



+ Nếu tắt tiến trình thành công, giao diện sẽ hiện thị như sau:

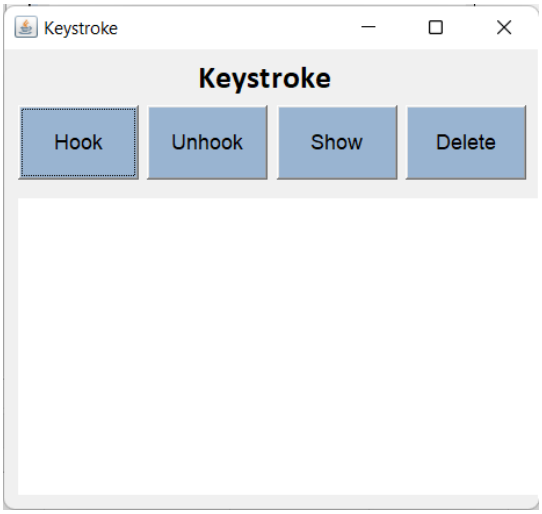


+ Nếu không tìm thấy tiến trình hoặc xảy ra lỗi khi tắt tiến trình, giao diện sẽ hiển thị như sau:



2.7. KeystrokeMenu.java

- Hàm setHookAction() dùng để gán vào hành động cần thực hiện là Hook nếu server chọn hành động Hook. Tương tự với các hàm setShowAction() và setUnhookAction().
- Hàm display() dùng để hiển thị cửa sổ Keystroke.
- Hàm addKeyText() dùng để nhận vào những ký tự đã nhận được.
- Hàm KeyStrokeMenu() hiển thị giao diện các lựa chọn cho server như : Hook, Unhook, Show, Delete.



2.8. Main.java

- Tạo UIManager để hiện thị lên form giao diện.
- Hàm startListening(): chạy một luồng mới dành cho việc lắng nghe thông tin từ server.
  + Nếu dữ liệu nhận được là ảnh (ImageIcon) thì gọi hàm addScreenShotImage() của UIManager để hiển thị ảnh ở ScreenShotMenu.
  + Nếu dữ liệu nhận được là mảng 2 chiều Object[][] thì gọi hàm displayProcess() của UIManager để hiển thị dữ liệu Process hoặc App Running ở ProcessMenu.
  + Nếu dữ liệu nhận được là chuỗi (String) thì kiểm tra:
    - Nếu chuỗi bắt đầu với "$-Key-$" thì gọi hàm addKeyLogText() của UIManager để hiển thị phím mà client nhấn.
    - Nếu chuỗi bắt đầu với "$-Successful-$", hoặc "$-Fail-$" thì thông báo cho người dùng việc thực thi của client thành công hay thất bại bằng hàm inform() của UIMananger.
    - Nếu là một chuỗi bình thường thì hiển thị tin nhắn từ client bằng hàm addMessage() của UIManager.
- Hàm addEvents(): cài đặt chức năng cho các nút trong giao diện của client bằng cách sử dụng lamda để gán giá trị cho các Consumer<String> của UIManager.

IV. TÀI LIỆU THAM KHẢO

- Java Robot: https://www.javatpoint.com/java-robot
- Java JFrame: https://www.javatpoint.com/java-jframe
- Java Socket: https://openplanning.net/10393/java-socket
- Process Handle: https://docs.oracle.com/javase/9/docs/api/java/lang/ProcessHandle.html
- Image: https://docs.oracle.com/javase/tutorial/2d/images/loadimage.html
- Consumer: https://docs.oracle.com/javase/8/docs/api/java/util/function/Consumer.html