MÔN: **PHÁT TRIỂN ỨNG DỤNG CÓ ĐỒ ÁN**

ĐỒ ÁN

ĐỀ TÀI: **CONTROLLING MACHINES REMOTELY WITH JAVA**

GVHD: Huỳnh Thái Học - Đỗ Hà Phương

NHÓM: 3T

TP.HCM-Tháng 08 Năm 2017

MÔN: **PHÁT TRIỂN ỨNG DỤNG CÓ ĐỒ ÁN**

ĐỒ ÁN

ĐỀ TÀI: **CONTROLLING MACHINES REMOTELY WITH JAVA**

GVHD: Huỳnh Thái Học - Đỗ Hà Phương

NHÓM: 3T

TP.HCM-Tháng 08 Năm 2017

THÀNH VIÊN

Nhóm Trưởng: Nguyễn Văn Chung
Sdt: 097060301
Mail: nguyenvanchung24864@gmail.com

Họ và Tên	MSSV	Ghi Chú
Nguyễn Văn Chơn	15052921	
Nguyễn Văn Chung	15087641	
Võ Thành Đạt	15070751	

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1. TỔNG QUAN

1.1 Giới thiều đề tài: Topic 6

Xây dựng chương trình“Controlling Machines Remotely with Java”.(Điều khiển từ xa).

Hiện nay, Công Nghệ Thông Tin (CNNT) đang phát triển mạnh mẽ và là ngành công nghiệp mũi nhọn ở nhiều quốc gia trên thế giới, trong đó có Việt Nam. Máy tính trở lên phổ biến, xuất hiện ngày càng nhiều ở các gia đình và trở thành một công cụ không thể thiếu của nhiều người.

Và với nhu cầu xử dụng máy tính hiện nay, con người không chỉ trực tiếp điều khiển trực tiếp trên chính máy tính, mà còn có nhu cầu điều khiên từ xa giống như các thiết bị điện tử khác (tivi, đầu đĩa, ...). Đó là một công cụ hỗ trợ đắc lực cho con người nhất là trong công việc giám sát hỗ trợ từ xa.

Trong đề tài này, chúng em thực hiện xây dựng một chương trình điều khiển từ xa, theo mô hình Client/Server, trong mạng LAN. Trong đó máy điều khiển là Server và máy bị điều khiển là Client. Sau khi kết nối thành công, máy Server sẽ có thể thực hiện các chức năng sau.

- **Chức năng yêu cầu**
  - Theo dõi , điều khiển màn hình máy Client.
- **Những chức năng mở rộng có thể làm thêm (nếu hoàn thành xong chức năng yêu cầu)**
  - Chia sẻ màn hình cho máy Client
  - Chụp màn hình máy Client
  - Truyền nhận file với máy Client
  - Thực hiện lệnh Shutdown, Restart máy Client

Ứng dụng sẽ được xây dựng bằng ngôn ngữ lập trình Java trên Eclipse.

1.2 Tiến độ thực hiện.

Tuần	Công việc	Thành viên	Ghi chú
Tuần 1 (15/8/2017)	- Hình dung, mô tả ứng dụng sẽ làm. - Lập kế hoạch cho 9 tuần tiếp theo.	- Nguyễn Văn Chơn - Nguyễn Văn Chung - Võ Thành Đạt	- **Hoàn thành (100%). Có file word**
Tuần 2 (22/8/2017)	- Tìm hiểu cách thiết lập mạng LAN trên VMWARE - Link cài Vmware: https://goo.gl/Zdyr24 - Link cài windown 7 trên Vmware: https://goo.gl/kx6kdd - Link Ping ip giữa 2 máy: https://goo.gl/5cU7gJ	- Nguyễn Văn Chơn - Nguyễn Văn Chung - Võ Thành Đạt	- **Hoàn thành (100%)**
	- Tiến hành xây dựng mạng LAN với 1 máy ảo và 1 máy thật	- Nguyễn Văn Chung	- **Hoàn thành (100%)**
Tuần 3 (29/8/2017)	- Tìm hiểu về lập trình Socket trên java.(xem code ví dụ). Link 1: https://goo.gl/4WNaSq Link 2: https://goo.gl/oYRE9T Link 3: https://goo.gl/6a628L	- Nguyễn Văn Chơn - Nguyễn Văn Chung - Võ Thành Đạt	- **Hoàn thành (100%)**
	- Làm theo ví dụ.	- Võ Thành Đạt - Nguyễn Văn Chơn	- **Hoàn thành (70%)**
Tuần 4 (05/9/2017)	- Tiến hành kết nối giữa 2 máy Client/Server (Trên mạng LAN đã	- Nguyễn Văn Chơn	- **Hoàn thành (100%).**

	thiết lập ở trên). Bằng câu lệnh trong java.	- Nguyễn Văn Chung - Võ Thành Đạt	**Nhưng bị lấn sang tuần 5.**
Tuần 5 (12/9/2017)	- Xem & hiểu cách theo dõi màn hình máy Client Link 1: https://goo.gl/oMNrnL Link 2: https://goo.gl/c8EaJL	- Nguyễn Văn Chơn - Nguyễn Văn Chung - Võ Thành Đạt	- **Hoàn Thành (100%)**
Tuần 6,7 (19/9/2017 & 26/9/2017)	- Hiện thực theo dõi màn hình máy Client	- Nguyễn Văn Chơn - Nguyễn Văn Chung - Võ Thành Đạt	- **Hoàn thành (100%)**
Tuần 8	- Thêm chức năng chat, shutdown (nếu có thể)	- Nguyễn Văn Chơn - Võ Thành Đạt	-
	- Hoàn thiện giao diện	- Nguyễn Văn Chung	-
Tuần 9&10	- Test & kiểm tra lỗi - Hoàn thiện ứng dụng (giao diện).	- Nguyễn Văn Chơn - Nguyễn Văn Chung - Võ Thành Đạt	

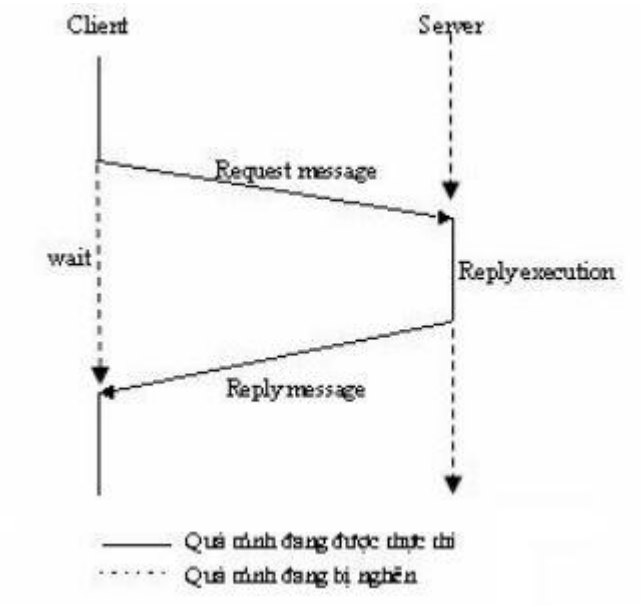
CHƯƠNG 2. THIẾT KẾ & CÀI ĐẶT ỨNG DỤNG

2.1 Cơ sở lý thuyết.

2.1.1 Giới thiệu mô hình Client-Server

Mô hình Client-Server được sử dụng trong các hệ phân tán và bao gồm hai thành phần riêng biệt là: Server đóng vai trò là người phục vụ, cung cấp chức năng và Client đóng vai trò là người tiêu thụ sử dụng chức năng đó. Client là

bên chủ động tạo kết nối và gửi yêu cầu đến Server, trong khi Server thụ động nắng nghe và thực hiện yêu cầu đó.



Hình 1. Mô hình Client-Server

2.1.1.1 Ưu điểm

Quản lý tập trung: dữ liệu được lưu trữ tập trung trên Server thay vì nằm rải rác trên nhiều máy, giúp đơn giản hóa việc truy xuất và cập nhật dữ liệu.

Dễ bảo trì: nhờ khả năng quản lý tập trung, vì vậy công việc bảo trì cũng dễ dàng và nhẹ nhàng hơn vì việc bảo trì phần lớn thực hiện trên Server

2.1.1.2 Cách thức hoạt động

Hoạt động theo kiểu giao thức bất đối xứng

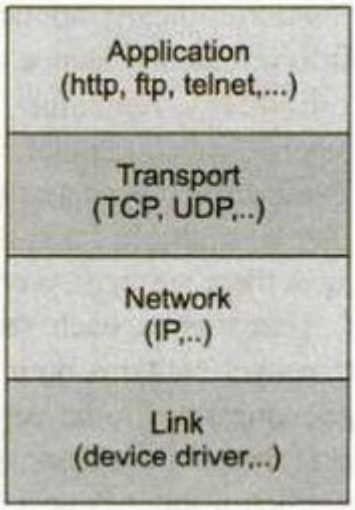
Thể hiện quan hệ một chiều giữa Client và một Server

Các tiến trình Client và Server có thể chạy cùng một host hoặc các host khác nhau và là các đối tượng logic tách biệt, liên lạc với nhau qua mạng để cùng thực hiện một công việc. Client bắt đầu phiên hội thoại bằng cách yêu cầu dịch vụ, Server sẵn sàng chờ các yêu cầu từ Client gửi tới.

2.1.2 Lập trình với giao thức TCP Socket

Trong mô hình Client-Server, thông thường các chương trình chạy trên máy Client sẻ gửi yêu cầu tới một chương trình (thường được gọi là chương

trình Server) đang chạy trên máy Server. Các gói yêu cầu này bao gồm các dịch vụ mạng được cung cấp bởi tầng giao vận (Transport layer) trong mô hình mạng, thường được gọi là TCP/IpP (Transport Control Protocol/Internet Protocol). Tầng này bao gồm hai giao thức là TCP và UDP (User Datagram Protocol)



Hình 2. Tầng Transport với TCP/IP

Giao thức TCP là một trong các giao thức cốt lõi của giao thức TCP/IP. Sử dụng TCP, các máy được nối mạng bới nhau có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tín. Giao thức này đảm bảo chuyền dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự.

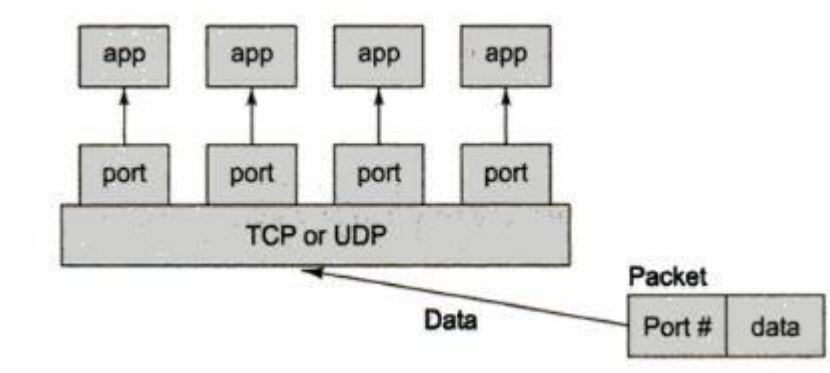
Giao thức UDP cũng là một trong các giao thức cốt lõi của giao thức TCP/IP. Sử dụng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thức tự truyền nhận. Các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo nào. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian.

2.1.2.1 Địa chỉ IP

Mọi máy tính trong môi trường mạng đều được xác định bởi địa chỉ IP 4-byte mà được viết dưới dạng chấm như 192.168.110.1, trong đó mỗi byte là một giá trị ký số từ 0 đến 255.

2.1.2.2 Cổng Port

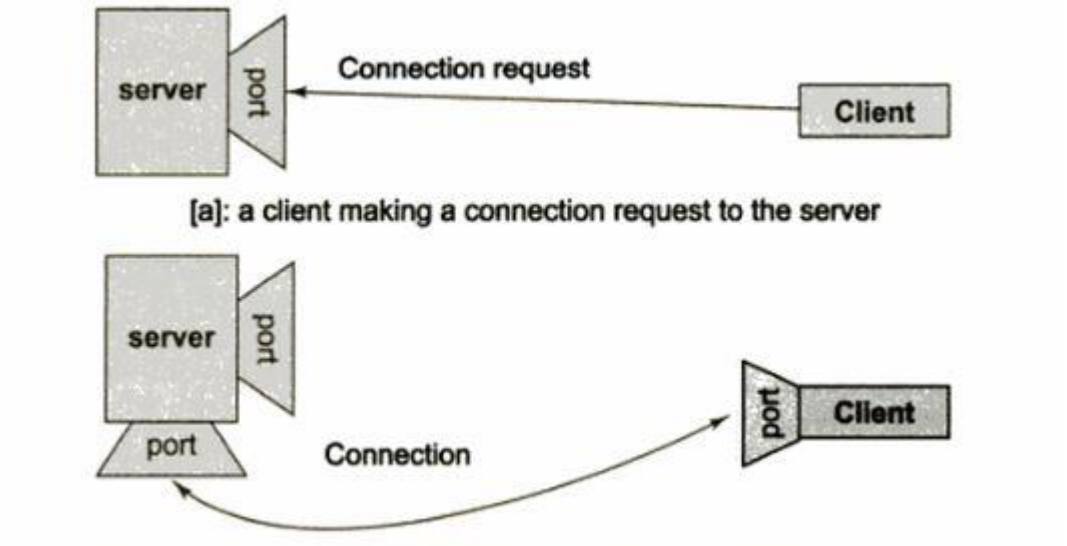
Thông thường, mỗi máy tính chỉ có một địa chỉ IP. Tuy nhiên các máy tính cần kết nối và cung cấp nhiều hơn một kiểu dịch vụ. Và để phân biệt các dịch vụ này ta có khái niệm về cổng (Port), là một điểm truy cập logic được biểu diễn bởi một số nguyên 16-bit gắn cho mỗi tiến trình mạng. Mỗi tiến trình mạng được gắn một cổng duy nhất.



Hình 3. Các gói tin được gán 1 cổng nhất định

- **Chấp nhận kết nối từ xa trên cổng đã được gắn.**

Một (chương trình) Server chạy trên một máy tính cụ thể và có một socket được gắn bởi một cổng cụ thể. Chương trình Server sẽ lắng nghe socket từ một Client để tạo một yêu cầu kết nối. Nếu thành công, chương trình server chấp nhận kết nối. Sau đó, chương trình server tạo ra một socket mới được gắn bởi một cổng khác và lắng nghe kết nối socket từ Client yêu cầu.



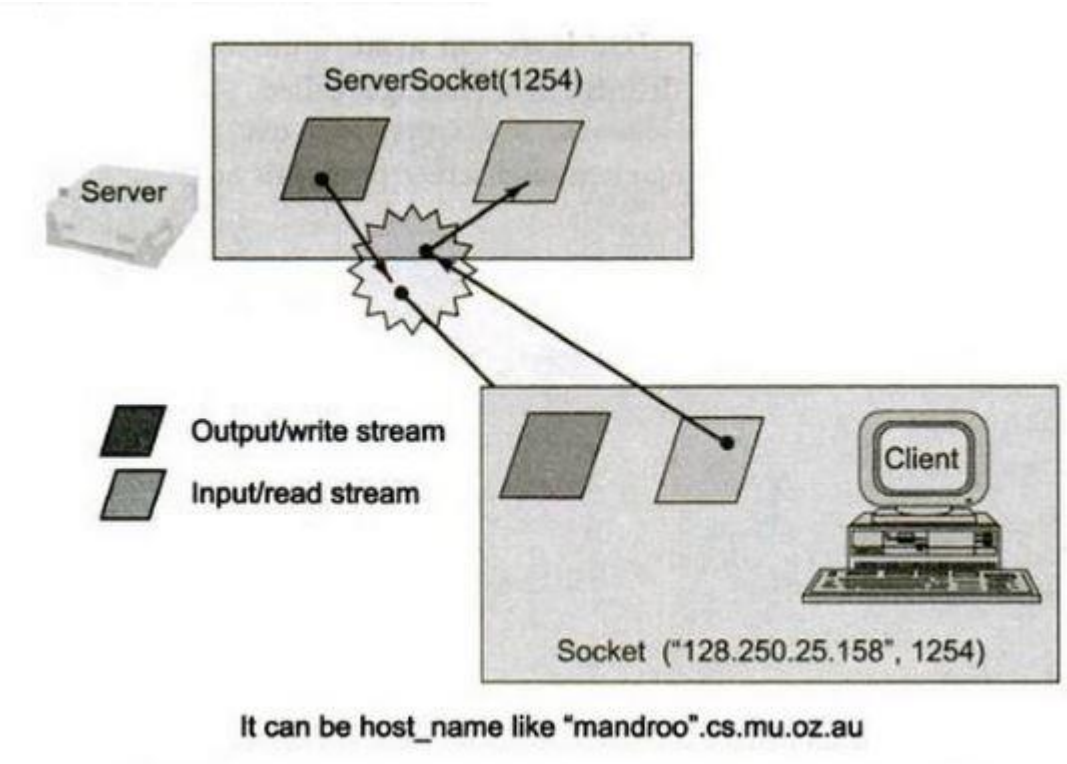
Hình 4. Thiết lập kết nối hai chiều giữa Client và Server

2.1.2.3 Lập trình TCP Socket

Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (Client) và một chương trình cung cấp dịch vụ (Server) trên mạng LAN, WAN hay Internet và đôi lúc là giữa những quá trình ngay bên trong máy tính. Mỗi socket có thể được xem như là một điểm cuối trong một kết nối. Khi một socket được thiết lập phù hợp, hai máy tính có thể trao đổi dịch vụ dữ liệu.

Có 2 lớp của gói java.net được dùng để tạo những chương trình server và client

Một chương trình server tạo một socket cụ thể được sử dụng để chờ lắng nghe kết nối từ phía client (server socket). Trong một yêu cầu kết nối, chương trình tạo ra một socket mới mà qua đó chương trình sẽ trao đổi dữ liệu với client sử dụng các luồng input và output. Người lập trình cần mở một socket, thực hiện việc tạo và biểu diễn các luồng input/output.



Hình 5. Chương trình client-server có dùng socket

2.2 Phương pháp triển khai.

Dùng phương pháp lập trình mạng TCP Socket threaded Server để tạo kết nối giữa máy Client với máy Server và tạo các luồng Input/Output.

Client sẽ kết nối tới Server thông qua địa chỉ IP của Server. Lúc này Server chấp nhận kết nối từ Client và thực hiện việc điều khiển.

Để cho phép nhiều client có thể kết nối đến Server thì Server phải là chương trình đa nhiệm, đa luồng cho phép nhiều tiến trình, tiểu trình có thể chạy song song cùng một thời điểm. Mỗi tuyến (thread) đảm nhận việc liên lạc với Client. Nghĩa là khi có một Client kết nối đến, chương trình Server sinh ra một tuyến (thread) để điều khiển việc truyền thông với Client.

2.3 Thiết kế ứng dụng

2.3.1 Chương trình trên Server (MachineServer)

- **Lớp ServerInitiator**

Đây là chờ các kết nối của khách hàng. Ngoài ra, nó tạo ra một phần thiết yếu của giao diện chương trình.

Code: Chờ Client kết nối

```
while(true){
    Socket client = sc.accept();
    System.out.println("Một máy Client mới kết nối");

    new ClientHandler(client,desktop);
}
```

- **Lớp ClientHandler**

Mỗi khách hàng kết nối, lớp này sẽ tạo ra một đối tượng (1 thread). Nó cho thấy khung màn hình của mỗi khách hàng và nhận được kích thước màn hình của khách hàng.

Code: Tạo giao diện cho mỗi Client khi kết nối đến

```
public void drawGUI(){
    interFrame.setLayout(new BorderLayout());
    interFrame.getContentPane().add(cPanel,BorderLayout.CENTER);
    interFrame.setSize(100,100);
    desktop.add(interFrame);
    try {
        interFrame.setMaximum(true);
    } catch (PropertyVetoException ex) {
        ex.printStackTrace();
    }
    cPanel.setFocusable(true);
    interFrame.setVisible(true);
}
```

- **Lớp ClientScreenReciever**

Nhận màn hình thu được từ máy khách, sau đó hiển thị nó.

Code: Nhận ảnh chụp và hiển thị chúng

```
while(continueLoop){

    ImageIcon imageIcon = (ImageIcon) cObjectInputStream.readObject();
    System.out.println("New image recieved");
    Image image = imageIcon.getImage();
    image = image.getScaledInstance(cPanel.getWidth(),cPanel.getHeight()
                                    ,Image.SCALE_FAST);

    Graphics graphics = cPanel.getGraphics();
    graphics.drawImage(image, 0, 0, cPanel.getWidth(),cPanel.getHeight(),cPanel);
}
```

- **Lớp ClientCommandsSender**

Chờ các lệnh của máy chủ, sau đó gửi chúng đến máy khách. Các lệnh của máy chủ bao gồm di chuyển chuột, nhấn bàn phím, nhấp chuột, v.v ...

Code: Nhận lệnh chuột và bàn phím, sau đó gửi chúng đến Client.

```
public void mouseDragged(MouseEvent e) {
}

public void mouseMoved(MouseEvent e) {
    double xScale = clientScreenDim.getWidth()/cPanel.getWidth();
    System.out.println("xScale: " + xScale);
    double yScale = clientScreenDim.getHeight()/cPanel.getHeight();
    System.out.println("yScale: " + yScale);
    System.out.println("Mouse Moved");
    writer.println(EnumCommands.MOVE_MOUSE.getAbbrev());
    writer.println((int)(e.getX() * xScale));
    writer.println((int)(e.getY() * yScale));
    writer.flush();
}

 public void mouseClicked(MouseEvent e) {
}

public void mousePressed(MouseEvent e) {
    System.out.println("Mouse Pressed");
    writer.println(EnumCommands.PRESS_MOUSE.getAbbrev());
    int button = e.getButton();
    int xButton = 16;
    if (button == 3) {
        xButton = 4;
    }
    writer.println(xButton);
    writer.flush();
}

public void mouseReleased(MouseEvent e) {
    System.out.println("Mouse Released");
    writer.println(EnumCommands.RELEASE_MOUSE.getAbbrev());
    int button = e.getButton();
    int xButton = 16;
    if (button == 3) {
        xButton = 4;
    }
    writer.println(xButton);
    writer.flush();
}
public void mouseEntered(MouseEvent e) {
}
public void mouseExited(MouseEvent e) {

}
public void keyTyped(KeyEvent e) {
}

public void keyPressed(KeyEvent e) {
    System.out.println("Key Pressed");
    writer.println(EnumCommands.PRESS_KEY.getAbbrev());
    writer.println(e.getKeyCode());
    writer.flush();
}

public void keyReleased(KeyEvent e) {
    System.out.println("Mouse Released");
    writer.println(EnumCommands.RELEASE_KEY.getAbbrev());
    writer.println(e.getKeyCode());
    writer.flush();
}
```

- **Lớp EnumCommands**

Xác định các hằng số được sử dụng để biểu diễn các lệnh máy chủ.

2.3.2 Chương trình trên Client (MachineClient)

- **Lớp ClientInitiator**

Đây là lớp thiết lập kết nối tới Server, thông qua địa chỉ IP của máy Server và tạo giao diện phía Client.

Code: Nhập địa chỉ IP và kết nối tới Server

```
public static void main(String[] args){
    String ip = JOptionPane.showInputDialog("Nhập địa chỉ IP Máy Server.");
    int port = 2345;
    new ClientInitiator().initialize(ip, port);
}

public void initialize(String ip, int port ){

    Robot robot = null;
    Rectangle rectangle = null;

    try {
        System.out.println("Connecting to server .........");
        socket = new Socket(ip, port);
        System.out.println("Connection Established.");
        GraphicsEnvironment gEnv=GraphicsEnvironment.getLocalGraphicsEnvironment();
        GraphicsDevice gDev=gEnv.getDefaultScreenDevice();
        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        rectangle = new Rectangle(dim);
        robot = new Robot(gDev);
        drawGUI();
          new ScreenSpyer(socket,robot,rectangle);
         new ServerDelegate(socket,robot);
    } catch (UnknownHostException ex) {
        ex.printStackTrace();
    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (AWTException ex) {
            ex.printStackTrace();
    }
}
```

- **Lớp ScreenSpayer**

Chụp màn hình theo định kỳ và gửi nó xuyên suốt tới máy Server

Code: Chụp ảnh màn hình và gửi chúng đến Server

```
while(continueLoop){

    BufferedImage image = robot.createScreenCapture(rectangle);

    ImageIcon imageIcon = new ImageIcon(image);

    try {
        System.out.println("Before sending image");
        oos.writeObject(imageIcon);
        oos.reset(); //Clear ObjectOutputStream cache
        System.out.println("New screenshot sent");
    } catch (IOException ex) {
       ex.printStackTrace();
    }
    try{
        Thread.sleep(100);
    }catch(InterruptedException e){
        e.printStackTrace();
    }
}
```

- **Lớp ServerDelegate**

Nhận lệnh ở máy Server và thực thi chúng ngay trên chính Client.

Code: Nhận lệnh từ Server và gọi phương thức robot để thực hiện các lệnh này.

```
public void run(){
    Scanner scanner = null;
    try {

        System.out.println("Preparing InputStream");
        scanner = new Scanner(socket.getInputStream());

        while(continueLoop){

            System.out.println("Waiting for command");
            int command = scanner.nextInt();
            System.out.println("New command: " + command);
            switch(command){
                case -1:
                    robot.mousePress(scanner.nextInt());
                break;
                case -2:
                    robot.mouseRelease(scanner.nextInt());
                break;
                case -3:
                    robot.keyPress(scanner.nextInt());
                break;
                case -4:
                    robot.keyRelease(scanner.nextInt());
                break;
                case -5:
                    robot.mouseMove(scanner.nextInt(), scanner.nextInt());
                break;
            }
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

- **Lớp EnumCommands**

Xác định các hằng số được sử dụng để biểu diễn các lệnh của máy Server.

2.4 Kết quả chạy chương trình

2.4.1 Phía Server

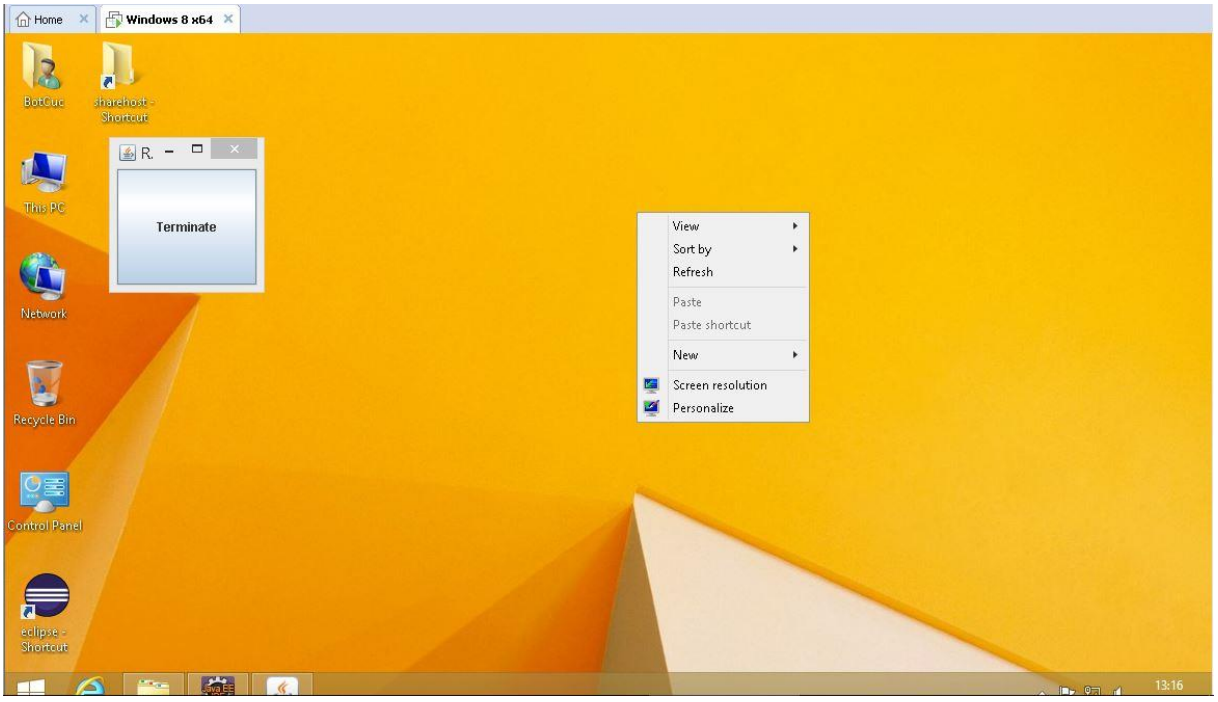
- **Khi nhấn chạy**

- **Sau khi nhấn "OK". Đang chờ Client kết nối tới**

2.4.2 Phía Client

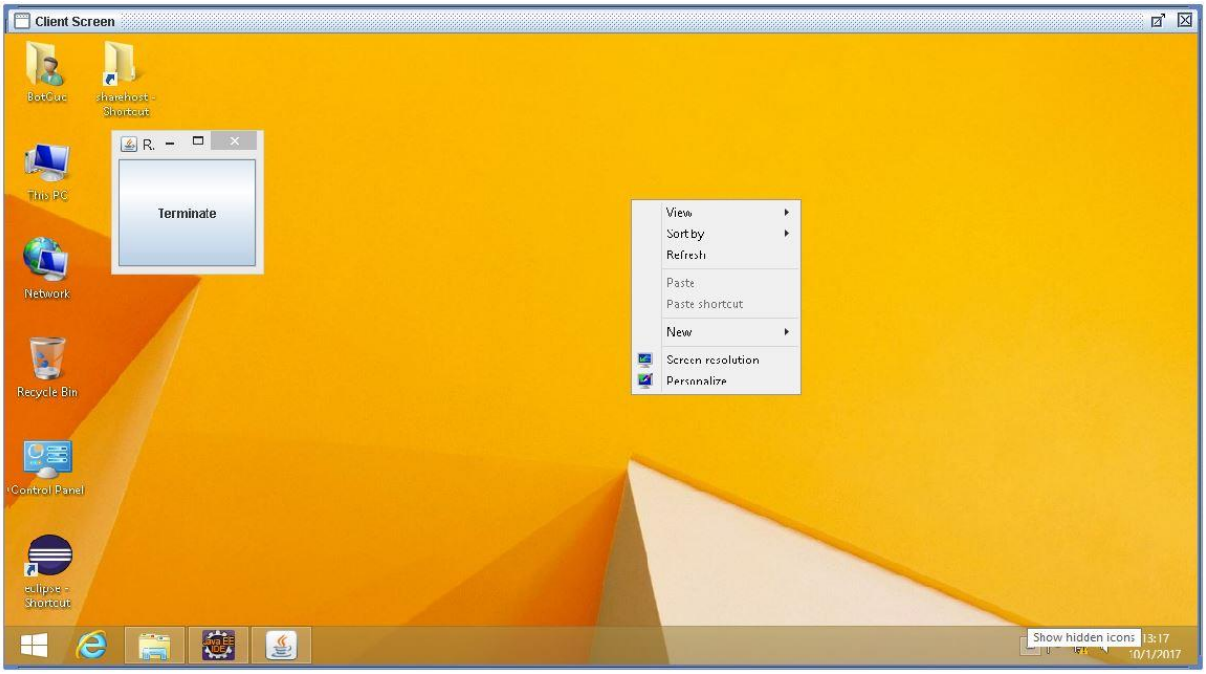
- **Khi nhấn chạy**

- **Sau khi nhấn "OK". Màn hình Win 8**



2.4.3 Kết quả:

Server nhận được màn hình Client.



CHƯƠNG 3. KẾT LUẬN

3.1 Nhận xét

Khối lương công việc hiện đã đạt được mục tiêu, yêu cầu của Giảng Viên đưa ra là xem và điều khiển được màn hình máy khác. Nhưng chưa hoàn thiện giao diện ứng dụng. Vậy nên đã hoàn thành 70% công việc.

3.2 Những công việc cần làm.

- **Hoàn thiện giao diện**
- **Thêm chức năng mở rộng-nếu được (Ví dụ: Chat, shutdown)**

3.3 Kế hoạch tiếp theo.

- **Nghiên cứu thêm về tạo giao diện trên java**
- **Nghiên cứu làm thế nào để gửi lệnh shutdown đến Client.**

3.4 Link GitHub

https://goo.gl/8cH9by

CHƯƠNG 4. TÀI LIỆU THAM KHẢO

1. Sách: Object-oriented progamming with Java: Essentials and applications
   Link: https://goo.gl/UW3S4m
2. Tìm hiểu về lập trình Socket trên java..
   Link 1: https://goo.gl/4WNaSq
   Link 2: https://goo.gl/oYRE9T
   Link 3: https://goo.gl/6a628L
3. Các chương trình mẫu tham khảo.
   Link 1: https://goo.gl/oMNrnL
   Link 2: https://goo.gl/c8EaJL