Báo cáo mô tả các hàm/thủ tục của chương trình, các giao diện

MỤC LỤC

Nội dung

1. Các thư viện được sử dụng.

Client	Server
import java.awt.image.BufferedImage; import java.io.BufferedInputStream; import java.io.BufferedReader; import java.io.File; import java.io.FileOutputStream; import java.io.FileWriter; import java.io.IOException; import java.io.InputStream; import java.io.InputStreamReader; import java.io.PrintStream; import java.net.Socket; import java.net.UnknownHostException; import java.util.Scanner; import javax.imageio.ImageIO; import java.io.IOException; import java.io.ObjectInputStream; import java.io.ObjectOutputStream; import java.net.Socket;	import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader; import java.io.PrintStream; import java.net.ServerSocket; import java.net.Socket; import java.util.ArrayList; import java.util.Scanner; import java.awt.AWTException; import java.awt.Dimension; import java.awt.Rectangle; import java.awt.Robot; import java.awt.Toolkit; import java.awt.image.BufferedImage; import java.io.File; import java.io.FileInputStream; import javax.imageio.ImageIO; **import com.github.kwhat.jnativehook.GlobalScreen;** **import com.github.kwhat.jnativehook.NativeHookException;** **import com.github.kwhat.jnativehook.keyboard.NativeKeyEvent;** **import com.github.kwhat.jnativehook.keyboard.NativeKeyListener;**
	JNativeHook.jar (thư viện hỗ trợ để bắt bàn phím)

2. Các biến toàn cục

Client	Server
Socket socket ; PrintStream ps = null; BufferedReader br = null; InputStream inputstream= null; static String keylog = ""; public final static int SERVER_PORT = 8888;	public final static int SERVER_PORT = 8888; static ArrayList<String> list = new ArrayList<String>(); static int demsolanchaythread = 1;

3. Kết nối Client với Server

Client	Server
public void connect(String SERVER_IP) { socket = new Socket(SERVER_IP, SERVER_PORT); ps = new PrintStream(socket.getOutputStream()); br = new BufferedReader(new InputStreamReader(socket.getInputStream())); }	public void server() { ServerSocket server = new ServerSocket(SERVER_PORT); Socket socket =server.accept(); BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream())); PrintStream ps = new PrintStream(socket.getOutputStream()); }
Kết nối socket và thiết lập các lớp gửi và nhận.	Kết nối socket và thiết lập các lớp gửi và nhận.

4. Cách thức hoạt động

<table>
<tr><th>Client</th><th>Server</th></tr>
<tr><td>

```
/*Ví dụ: khi ta ấn vào nút chụp màn hình: chương trình
sẽ chạy hàm sau*/

public void Screenshot() throws IOException
        {
                ps.println("1");
         // làm gì đó ở đây.
         ......
}
```

</td><td>

```
while (true) {
      int i = 0;
      String request = br.readLine();
      if(request != null)
          {
              i = Integer.parseInt(request);
            if(i == 1)
            {
                  screen(br, ps, socket);
            }
            else if(i==2)
            {
                  listprocess(br, ps);
            }
            …
            else if(i==100)
             {  break;   }
          }
}
```

</td></tr>
<tr><td>Khi ấn 1 nút bên Client thì Client sẽ gửi qua bên server 1 ký tự số từ 1 đến 100. Mỗi ký tự sẽ yêu cầu Server thực hiện 1 hàm đã cài đặt sẵn. Và Client cũng sẽ thực hiện các lệnh nhận lại dữ liệu khi ấn nút</td><td>Khi Server được bật thì nó sẽ khởi động 1 vòng while(true) và đợi dữ liệu từ Client .Khi nhận được ký tự từ Client, Server sẽ đưa đưa đi so sánh nếu đúng ký tự nào nó sẽ thực hiện câu lệnh đó</td></tr>
</table>

5. Chụp màn hình

<table>
<tr><th>Client</th><th>Server</th></tr>
<tr><td>

```
public void Screenshot() {
ps.println("1");

      FileOutputStream fout = new
FileOutputStream("screen.jpg");
      int j;
      while (true)
      {
         String r = br.readLine();
           j = Integer.parseInt(r);
         if(j <= -1)
         {
           break;
         }
         fout.write(j);
      }
      fout.flush();
      fout.close();
}
```

</td><td>

```
public void screen(BufferedReader br, PrintStream ps) {
           String outFileName = "screen.jpg";
           Toolkit toolkit = Toolkit.getDefaultToolkit();
           Dimension screenSize = toolkit.getScreenSize();
           Rectangle screenRect = new Rectangle(screenSize);
           Robot robot = new Robot();
           BufferedImage image =
robot.createScreenCapture(screenRect);
           ImageIO.write(image, "jpg", new File(outFileName));
           //===============================================
           int j;
              FileInputStream fis = new FileInputStream ("screen.jpg");
               while ((j = fis.read()) > -1)   {
                        ps.println(j);
                    }
        ps.println(j);
              fis.close();
           }
```

</td></tr>
<tr><td>Khi ta ấn vào nút chụp mành hình thì hàm trên được thực hiện. Đầu tiên Client sẽ gửi ký tự “1” qua bên Server. Sau đó thực hiện bắt file mà Server gửi qua. Sau khi nhận được dữ liệu thì ghi xuống file screen.jpg.</td><td>Khi nhận được được ký tự “-1” từ Client thì Server sẽ thực hiện hàm trên. Đầu tiên máy tính Server sẽ chụp màn hình lại và ghi xuống file screen.jpg. Sau đó sẽ gửi file này qua bên Client.</td></tr>
<tr><th colspan="2">Giao Diện Client</th></tr>
<tr><td>

// Nút Chụp Hình

```
JButton ButtonScreenshot = new
JButton("Screenshot",snapIcon);
           ButtonScreenshot.addActionListener(new
ActionListener() {
public void actionPerformed(ActionEvent e){
  c.Screenshot();      // c này là đối tượng Client c;
  screenshot scrs = null; // screenshot là Jframe
scrs = new screenshot(c);
scrs.setVisible(true);
}
});
```

</td><td>

// Nút Lưu File

```
JButton Buttonsave = new JButton("SAVE",saveIcon);
Buttonsave.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
 File file = null;
JFileChooser fileChooser = new JFileChooser();
Component modalToComponent = null;
if (fileChooser.showSaveDialog(modalToComponent) ==
JFileChooser.APPROVE_OPTION) {
           file = new File(fileChooser.getSelectedFile() + ".jpg");
           fileChooser.setDialogTitle("Save Screenshot");
}
        ImageIO.write(image, "jpg", file);
}
});
```

</td></tr>
</table>

Khi bấm nút “Screenshot” thì chương trình sẽ mở một Jframe **screenshot**. Jframe này sẽ đọc file screen.jpg lên màn hình. Và tạo ra 2 nút bấm. Tiếp tục chụp hoặc lưu file xuống 1 vị trí trong máy tính.	Nút tiếp tục chụp sẽ gọi mới 1 Jframe **screenshot.** Nút lưu sẽ cho chúng ta chon vị trí để luu file.

6. Đọc danh sách Process

Client	Server
public void ListProcess { ps.println("2"); int j; FileWriter fw = new FileWriter("process.txt"); while (true) { String r = br.readLine(); j = Integer.parseInt(r); if(j == -1) { break; } fw.write(r+ "\n"); }	public void listprocess(BufferedReader br, PrintStream ps) { String line; @SuppressWarnings("deprecation") Process p = Runtime.getRuntime().exec (System.getenv("windir") +"\\system32\\"+"tasklist.exe"); BufferedReader input = new BufferedReader(new InputStreamReader(p.getInputStream())); while ((line = input.readLine()) != null) { ps.println(line); } line= "-1"; ps.println(line); input.close(); }
Đầu tiên Client sẽ gửi đến Server ký tự “2”. Sau đó sẽ kiểm tra chuỗi gửi đến có phải là ký tự “-1” hay không. Nếu là ký tự “-1” thì sẽ thoát vòng lặp và không chờ đọc dữ liệu nữa. Sau khi đọc xong thì sẽ lưu dữ liệu xuống file process.txt	Khi nhận được ký tự “2” từ Client, Server sẽ thực hiện lấy danh sách các process rồi gửi từ dòng qua cho Client. Khi không còn dữ liệu để gửi nữa thì sẽ gửi ký tự “-1” để cho Client biết kết thúc việc nhận dữ liệu.
Giao Diện Client: Khi Bấm nút "VIEW PROCESS"	
Object [][] tableData = {}; DefaultTableModel model;	JButton buttonviewlistprocess = new JButton("VIEW",viewIcon); buttonviewlistprocess.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { c.ListProcess(); **// gọi hàm ListProcess() ở trên** File file = new File("process.txt"); BufferedReader brf = new BufferedReader(new FileReader(file));

String [] tableCols = {"Process Name","Process ID"}; model = new DefaultTableModel(tableData,tableCols) { public boolean isCellEditable(int row, int column) { return false; } };	String k ; int dem=0; while ((k = brf.readLine()) != null) { if(dem > 2) { **// bỏ 3 dòng đầu của file process.txt** k = k.trim();**// bỏ các khoảng trắng đauf, cuổi chuỗi** String [] words=k.split("\\s+"); **//Cắt chuỗi** if(6 == words.length) { Object[] newobj = {words[0],words[1]}; **//Chỉ lấy tên và ID** model.addRow(newobj); } **//trường hợp tên process nhiều hơn 1 từ. vd: System Idle Process** Else { int temp = words.length; String NameProcess = ""; for(int i=0;i<temp - 5;i++){ NameProcess = NameProcess+ words[i] + " "; } Object[] newobj = {NameProcess,words[temp - 5]}; model.addRow(newobj); } } dem++; } } });
	Khi bấm nút view thì sẽ mở file "process.txt" và đọc từ dòng dữ liệu. Và cắt chuỗi ra để lấy Tên và Id của process. Rồi sau đó đưa vào 1 Object. Và đọc hiển thị lên 1 Jtable.

7. Kill Process

Client	Server
public void KillProcess(String PID) { ps.println("3"); ps.println(PID); }	public void Killprocess(BufferedReader br, PrintStream ps) { String r = br.readLine(); String k = "taskkill /PID " + r; Process builder = new ProcessBuilder("cmd.exe", "/c", k).start(); }
Khi nhấn nút Kill Process. Thì Client sẽ gửi đi ký tự "3" đến Server. Rồi tiếp tục gửi ID của process mà mình muốn kill	Khi Server nhận được ký tự "3" thì Server sẽ thực hiện hàm trên. Server sẽ đợi để nhận ID của process rồi thực hiện lệnh kill.

8. Start Process

Client	Server
public void StartProcess(String NameProcess) { ps.println("4"); NameProcess = NameProcess + ".exe"; ps.println(NameProcess); }	public void Startprocess(BufferedReader br, PrintStream ps) { String k = br.readLine(); Process process = new ProcessBuilder(k).start(); }
Khi nhấn nút Start Process. Thì Client sẽ gửi đi ký tự "4" đến Server. Rồi tiếp tục gửi Tên của process mà mình muốn Start	Khi Server nhận được ký tự "4" thì Server sẽ thực hiện hàm trên. Server sẽ đợi để nhận Tên của process rồi thực hiện lệnh Start.

9. Đọc danh sách App

Client	Server
public void ListApp() { ps.println("6"); int j; FileWriter fw = new FileWriter("applications.txt"); while (true) { String r = br.readLine(); j = Integer.parseInt(r); if(j == -1) { break; } fw.write(r+ "\n"); } fw.close(); }	public void listapp(BufferedReader br, PrintStream ps) { Process process = new ProcessBuilder("powershell","\"gps| ? {$_.mainwindowtitle.length -ne 0} | Format-Table -HideTableHeaders name, ID").start(); new Thread(() -> { Scanner sc = new Scanner(process.getInputStream()); if (sc.hasNextLine()) sc.nextLine(); String line; while (sc.hasNextLine()) { line = sc.nextLine(); ps.println(line); } line = "-1"; ps.println(line); }).start(); process.waitFor(); }
Đầu tiên Client sẽ gửi đến Server ký tự "6". Sau đó sẽ kiểm tra chuỗi gửi đến có phải là ký tự "-1" hay không. Nếu là ký tự "-1" thì sẽ thoát vòng lặp và không chờ đọc dữ liệu nữa. Sau khi đọc xong thì sẽ lưu dữ liệu xuống file applications.txt	Khi nhận được ký tự "6" từ Client, Server sẽ thực hiện lấy danh sách các process rồi gửi từ dòng qua cho Client. Khi không còn dữ liệu để gửi nữa thì sẽ gửi ký tự "-1" để cho Client biết kết thúc việc nhận dữ liệu.

Giao Diện Client: Khi Bấm nút "VIEW APP"	
Object [][] tableData = {}; DefaultTableModel model1; String [] tableCols = {"Application Name","Application ID"}; model1 = new DefaultTableModel(tableData,tableCols) { public boolean isCellEditable(int row, int column){ return false; } };	JButton ButtonViewApp = new JButton("VIEW",viewIcon); ButtonViewApp.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { c.ListApp();**// gọi hàm ListApp() ở trên** File file = new File("applications.txt"); BufferedReader brf= new BufferedReader(new FileReader(file)); String k ; while ((k = brf.readLine()) != null) { k = k.trim(); String [] words=k.split("\\s+"); Object[] newobj = {words[0],words[1]};

	//Chỉ lấy tên và ID model1.addRow(newobj); } } });
	Khi bấm nút view thì sẽ mở file “applications.txt” và đọc từ dòng dữ liệu. Và cắt chuỗi ra để lấy Tên và Id của app. Rồi sau đó đưa vào 1 Object. Và đọc hiển thị lên 1 Jtable.

10. Kill App

Client	Server
public void KillApp(String Aid) { ps.println("7"); ps.println(Aid); }	public void Killapp(BufferedReader br, PrintStream ps) { String r = br.readLine(); String k = "taskkill /PID " +r; Process builder = new ProcessBuilder("cmd.exe", "/c", k).start(); }
Khi nhấn nút Kill App. Thì Client sẽ gửi đi ký tự “7” đến Server. Rồi tiếp tục gửi ID của App mà mình muốn kill	Khi Server nhận được ký tự “7” thì Server sẽ thực hiện hàm trên. Server sẽ đợi để nhận ID của App rồi thực hiện lệnh kill.

11. Start App

Client	Server
public void StartApp(String NameApp) { // start a app ps.println("8"); ps.println(NameApp); }	public void Startapp(BufferedReader br, PrintStream ps) { String k = br.readLine(); k = "start " + k; Process builder = new ProcessBuilder("cmd.exe", "/c", k).start(); }
Khi nhấn nút Start App. Thì Client sẽ gửi đi ký tự “8” đến Server. Rồi tiếp tục gửi Tên của App mà mình muốn Start	Khi Server nhận được ký tự “8” thì Server sẽ thực hiện hàm trên. Server sẽ đợi để nhận Tên của App rồi thực hiện lệnh Start.

12. Nút Xóa (giao diện Client)

Process	APP
JButton btnNewButton = new JButton("CLEAR",delappIcon); btnNewButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { model.setRowCount(0); } });	JButton ButtonClear = new JButton("CLEAR",delappIcon); ButtonClear.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { model1.setRowCount(0); } });
Khi ấn vào nút “CLEAR “ thì chỉ việc thực hiện lệnh làm Jtable rỗng.	Khi ấn vào nút “CLEAR” thì chỉ việc thực hiện lệnh làm Jtable rỗng.

13. Keystroker

- Hook và Unhook

Client	Server
public void Hook() { ps.println("10"); } public void UnHook() { ps.println("11"); keylog = br.readLine(); **//String keylog là biến của lớp đã khai báo** }	static Thread thread = new Thread() { public void run() { try{ GlobalScreen.registerNativeHook(); GlobalScreen.addNativeKeyListener(new NativeKeyListener(){ @Override public void nativeKeyTyped(NativeKeyEvent nativeEvent){} @Override public void nativeKeyReleased(NativeKeyEvent nativeEvent){ String keyText; keyText=NativeKeyEvent.getKeyText(nativeEvent.getKeyCode()).toLowerCase(); list.add(keyText); } public void nativeKeyPressed(NativeKeyEvent nativeEvent){} }); } catch (NativeHookException e){ e.printStackTrace(); } } }; public void keylog(BufferedReader br, PrintStream ps) { list.clear(); list=new ArrayList<String>(); if(demsolanchaythread == 1) { thread.start(); **// chạy thread phía trên nếu nó chưa kích hoạt lần đầu** demsolanchaythread++; } String keystroke = br.readLine(); if(Integer.parseInt(keystroke) == 11) { String listString = new String(""); for (String a : list){ listString += a ; } **//ghép các từ lại thành 1 String để gửi nó đi** ps.println(listString); ps.flush(); }}

Khi ấn nút “hook” bên Client thì Client sẽ gửi ký tự “10” đến Server và yêu cầu bắt bàn phím. Khi ấn nút “Unhook” thì Client sẽ gửi ký tự “11” đến Server yêu cầu gửi các lại chuỗi ký tự đã bắt được.	Khi nhận được ký tự “10” từ Client. Server sẽ xóa danh sách các ký tự đã bắt được, và tạo ra một danh sách mới rỗng. Và kiểm tra xem Thread bắt bàn phím đã chạy hay chưa? Nếu chưa chạy thì sẽ chạy thread bắt bàn phím. Sau khi chạy thread thì sẽ đợi lệnh Unhook từ Client. Khi Server nhận được ký tự “11” từ Server thì Server sẽ nối các danh sách ký tự lại thành 1 chuỗi và gửi nó cho Client.

- Print và Delete (giao diện Client)

Print	Delete
public String ViewKeyStroker() { String temp = keylog; keylog = ""; return temp; } JButton viewButton = new JButton("PRINT",printIcon); viewButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { String pre = textPaneKeylog.getText(); String keylognew = c.ViewKeyStroker(); textPaneKeylog.setText(pre + " " + keylognew); } });	JButton ClearButton = new JButton("DELETE",delIcon); ClearButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { textPaneKeylog.setText(null); } });
Khi ta ấn nút “View” thì hàm **ViewKeyStroker()** được thực hiện và trả ra chuỗi đã nhận được từ Server. Client sẽ cộng chuỗi cũ đã hiển thị trên JtextPane và chuỗi mới lại. Và sẽ hiển thị lại chuỗi lên JtextPane.	Khi ta ấn nút xóa thì chỉ cần thực hiện lệnh làm cho JtextPane rỗng.

14. Shutdown (tắt máy)

Client	Server
public void Shutdown() { ps.println("12"); **// Shutting down the PC after 10 seconds**. }	public void Shutdown() { Runtime runtime = Runtime.getRuntime(); try { runtime.exec("shutdown -s -t 10"); } catch(IOException e) { System.out.println("Exception: " +e); } }
Khi ấn nút "Shutdown" thì Client sẽ gửi ký tự "12" đến Server để thực hiện lệnh Shutdown.	Khi nhận được ký tự "12". Server sẽ thực hiện hàm **Shutdown()** trên. Và máy Server sẽ tắt sau 10 giây.

15. Exit (thoát)

Client	Server
public void ExitSocket() { ps.println("100"); **//thoat chuong trinh** }	else if(i==100) { socket.close(); System.exit(0); break; }
Khi ấn nút "Exit" thì Client sẽ gửi ký tự "100" đến Server để thực hiện lệnh ngắt kết nối socket và thoát khỏi chương trình.	Khi nhận được ký tự "100". Server sẽ thực hiện việc ngắt kết nối và thoát khỏi chương trình.

16. Tài liệu khao thảo

- stackoverflow.com
- geeksforgeeks.org
- github.com
- demo của Giáo viên