CNLine Final Project Report

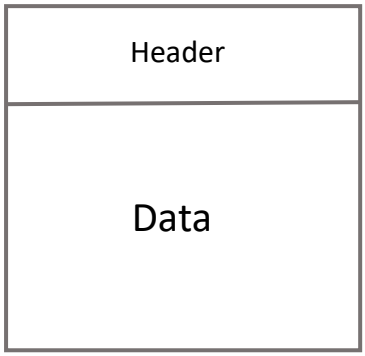
Content Page

Group3 : b02502129 黃柏瑋

t05902115 劉俊愷

b01501038 孫翌庭

1.為了本次期末 project，我們設計了如下 protocol

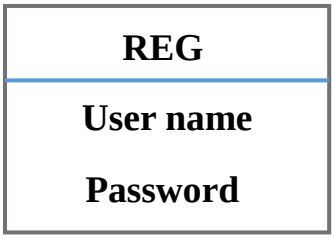


所有 Clinet 跟 Server 交換的訊息都是由一個 Header 與底下 Data 組成，Header 指出這個封包的 Type，DATA 指出剩下的資料

（1）Register

具體而言，對各個功能模組，Client 與 Server 傳輸的 Message 如下

From client to server

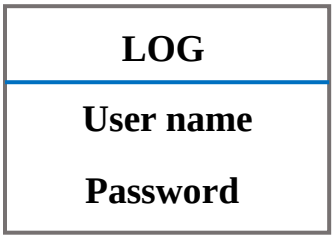


From server to client

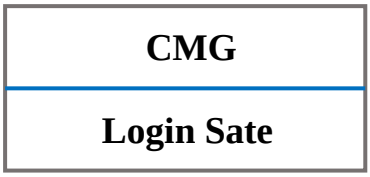
Register Sate:
Success (註冊成功)
Exist (帳號已存在)

（2）Login:

From client to server



From server to client

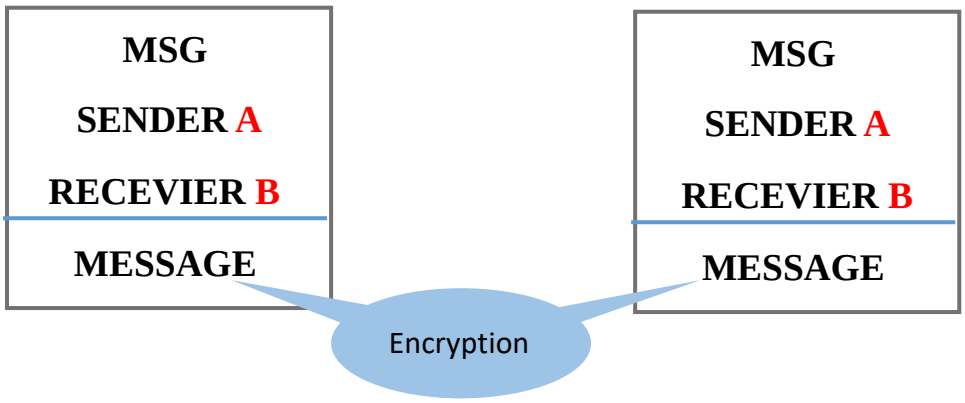


2 Login Sate:
Success (登入成功)
Repetition (重複登入)
Failed (帳號密碼錯誤)

（3）Message

From client to server

From server to client



（4）File transfer:

From client A to Server

From Server to client A

From Server to client B

FLIE
2
A

Two more Protocol

GRP	CMG
創群聊訊息	Offline (離線請求)

2. Function Specifictions:

（1）Register:
允許用戶註冊新帳號，但是會檢測帳戶是否存在。帳號用帳戶名來唯一標示。
（2）Login:
使用者輸入帳號密碼登入。
即時顯示上線及離線的使用者。
（3）Chat:
支援繁體中文、簡體中文、English 文字類別。選擇一位使用者或是聊天機器人，並傳送文字訊息。
所有的歷史對話紀錄都會保存在 Log File 裡，使用者可以隨時存取 Log File 查看過去的聊天記錄。
（4）Offline Message:
如果對離線的使用者傳送訊息，該位使用者在下次上線時可以接收到以傳送的訊息。
（5）Multiple File Transfer:
支援 JPG、JPEG 類型檔案傳送　//
Client 可以一次同時選擇多個檔案傳送給一位
使用者。
傳送進度顯示: 使用進度條的顯現方式，將檔案的傳送進度即時的顯示給使用者，傳輸完成時會發送提醒訊息給使用者。
（6）Bonus
密碼強度偵測:
根據密碼輸入的字元類型數目(英文、數字、符號)、不同類型文字是否交錯、以及密碼長度等評量項目，以紅色(弱、危險)、黃色(中等、試中)、綠色(強、良好)的顏色指示輸入的密法強度級別。
訊息加密 Encryption:
AES 加密方式對傳送的訊息進行加密；使用 WireShark 等封包擷取軟體獲取封包資訊時，會因為亂碼的顯示而無法對傳送的訊息或是密碼進行竊聽。
聊天機器人:
整合了網路上的聊天機器人系統，讓使用可以選擇對聊天機器人發送訊息，機器人會自動回覆訊息，增加讓使用者和聊天機器人對話的功能。
多人聊天功能:
使用者可以自行設定加入群組對話的其他使用者。在群組裡發送訊息，所以群組內的成員都會收到訊息，而群組之外的使用者

3.User & Operator Guide

Server GUI

開啟 Server: 按下上方視窗的 OPEN，即可以設定的 Port Number 開啟 Server。
設定 Port Number: 在 Port 的空格內可以輸入指定的 Server Port Number。
關閉 Server:按下上方視窗的 Close，即可關閉 Server。
儲存歷史紀錄: 按下 SAVE LOG，可以保存 Server 的歷史紀錄。

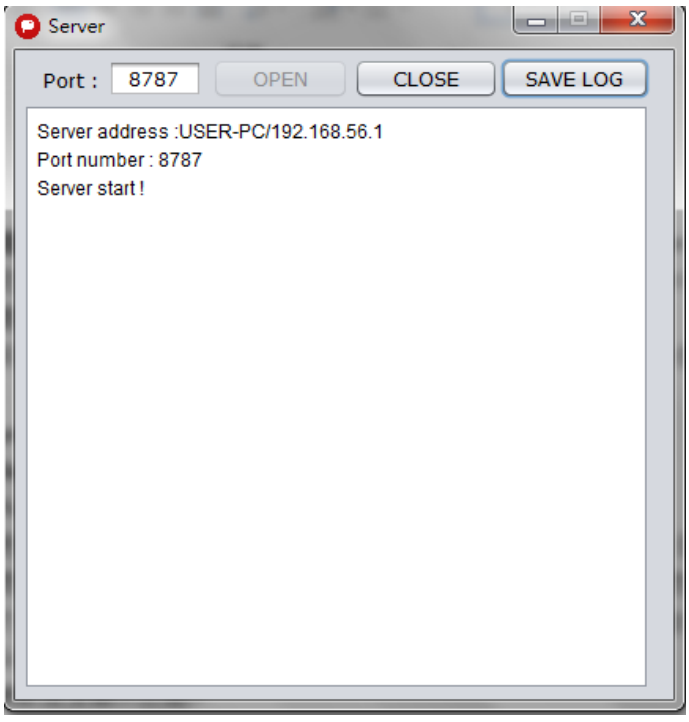
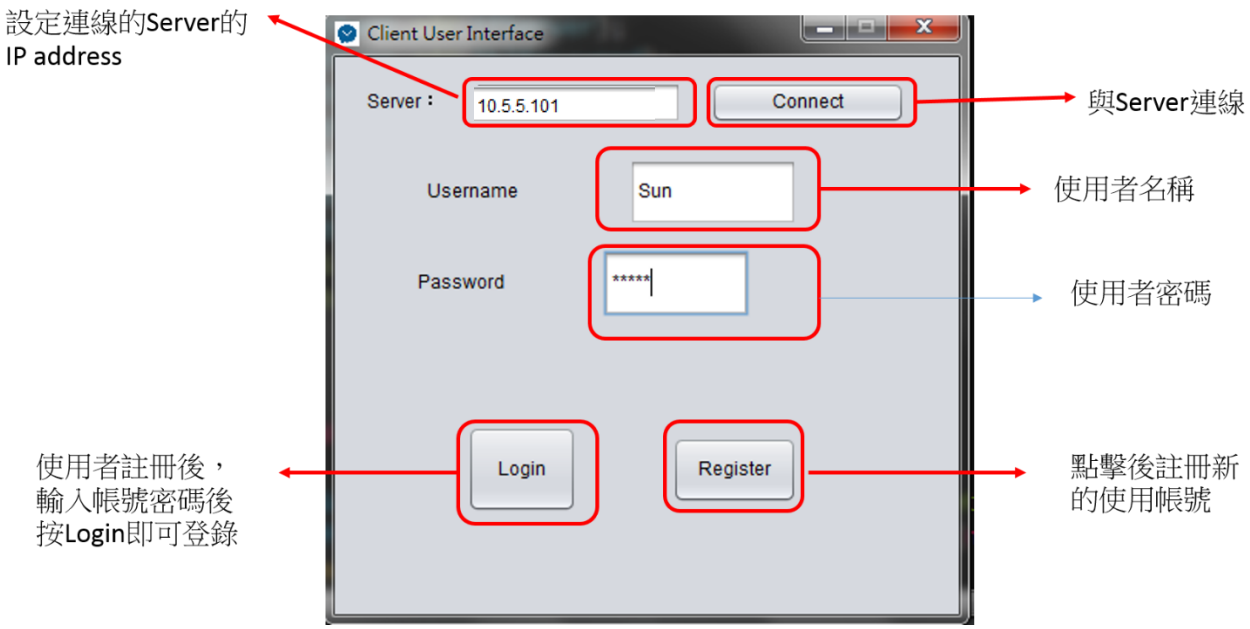


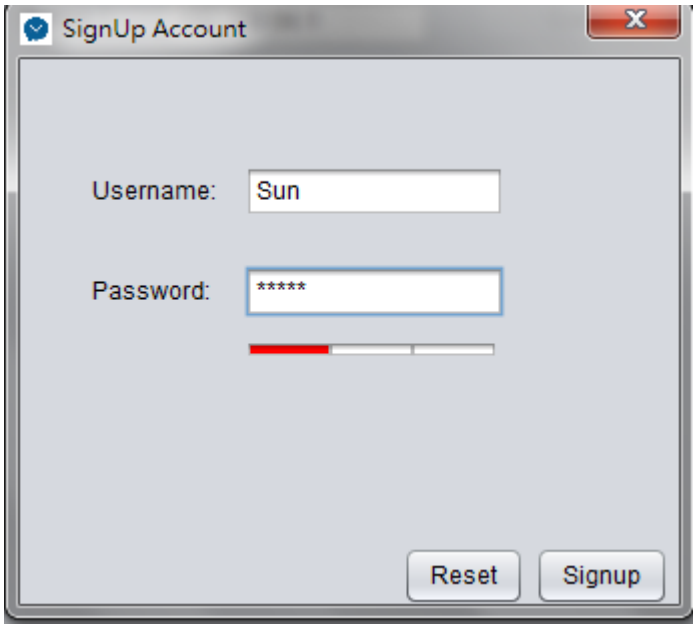
Fig: 開啟 Server 後的顯示訊息。

Client GUI

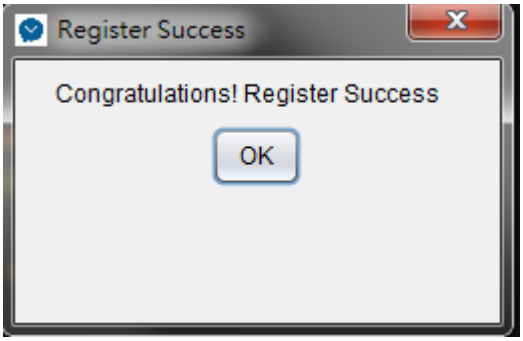


註冊:

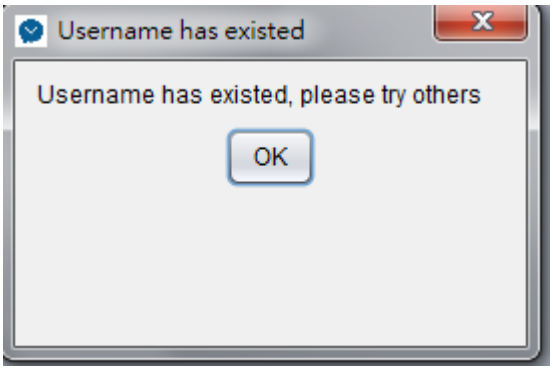
1. 按下 Register 進入註冊頁面。



2. 輸入新的帳號及密碼，密碼輸入下方有密碼強度提示。確認後按下 Signup。
   按下 Reset 可以清空已經輸入的資料。
3. 註冊成功會跳出提示訊息，即可回到原本頁面輸入新註冊的帳號寄碼，然後
   按下 Login 登入。

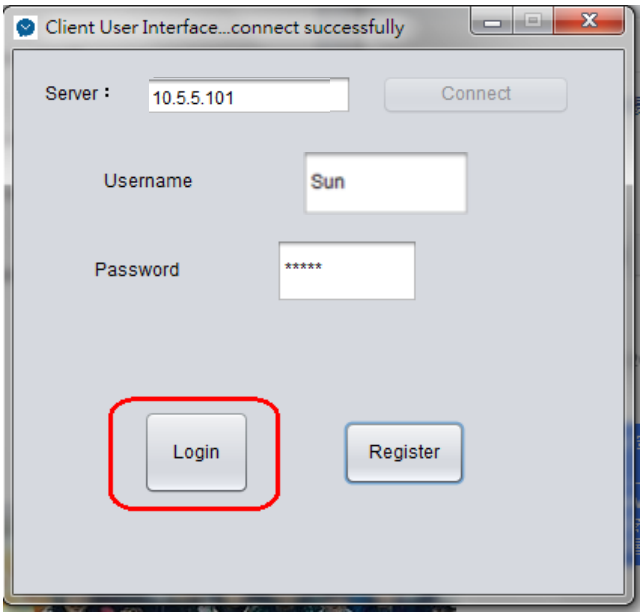


4. 如果輸入重複的帳號，系統會跳出提示訊息，請輸入新的帳號。

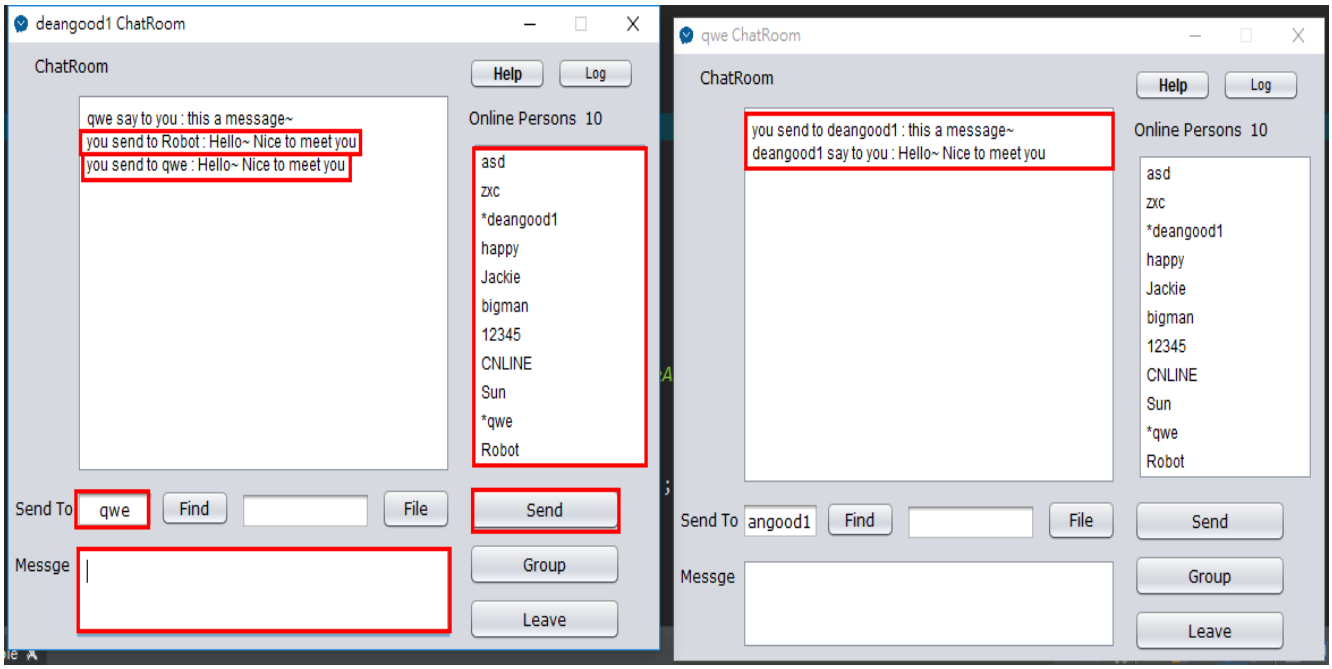


登錄:

1. 在起使頁面中入輸入已註冊的帳號及密碼後，按下 Login 即可成功登入。



聊天室 GUI



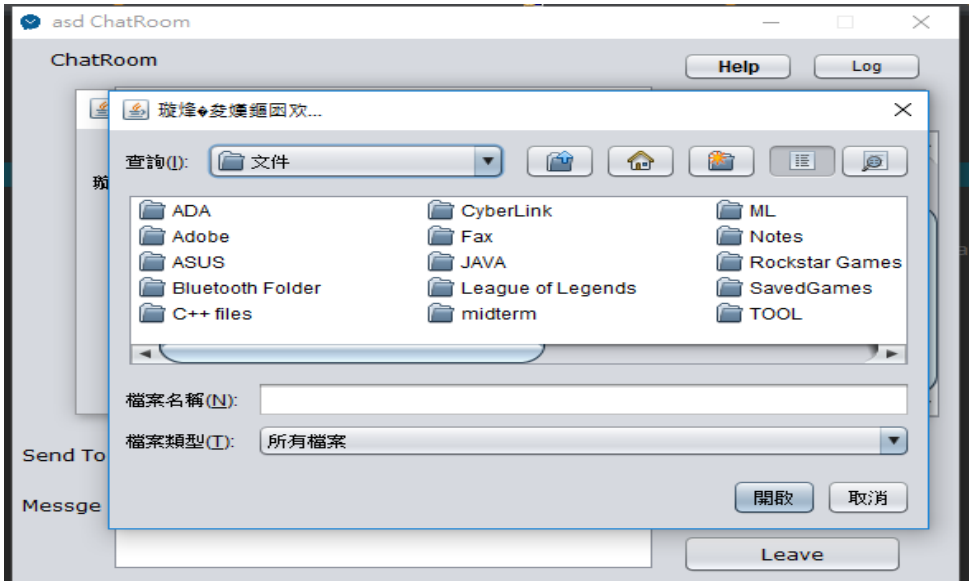
聊天功能:

發送訊息:

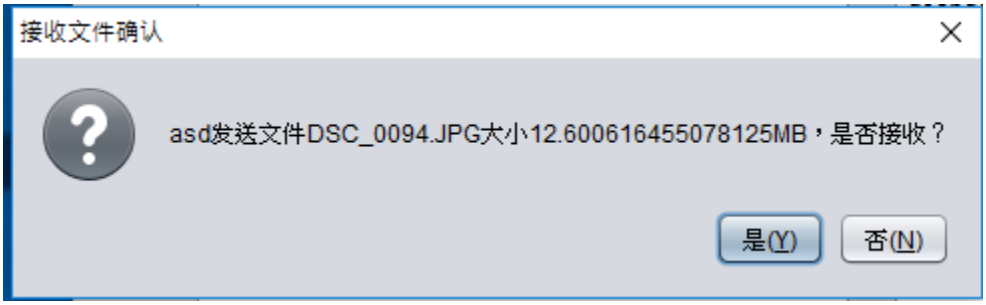
1. 點擊 GUI 右側的 User list，選擇要發送訊息的 User。然後在 Message 的方塊中輸入要送出的文字訊息，輸入完成後按下 Send 送出訊息。發送訊息和接收訊息範例如上圖所示。

傳送多份文件:

1. 選擇一位要發送文件的使用者
2. 點選 File。
3. 在本機系統中選擇多個要傳送的文件，確定後點選開啟，再按 Send，完成檔案傳送。

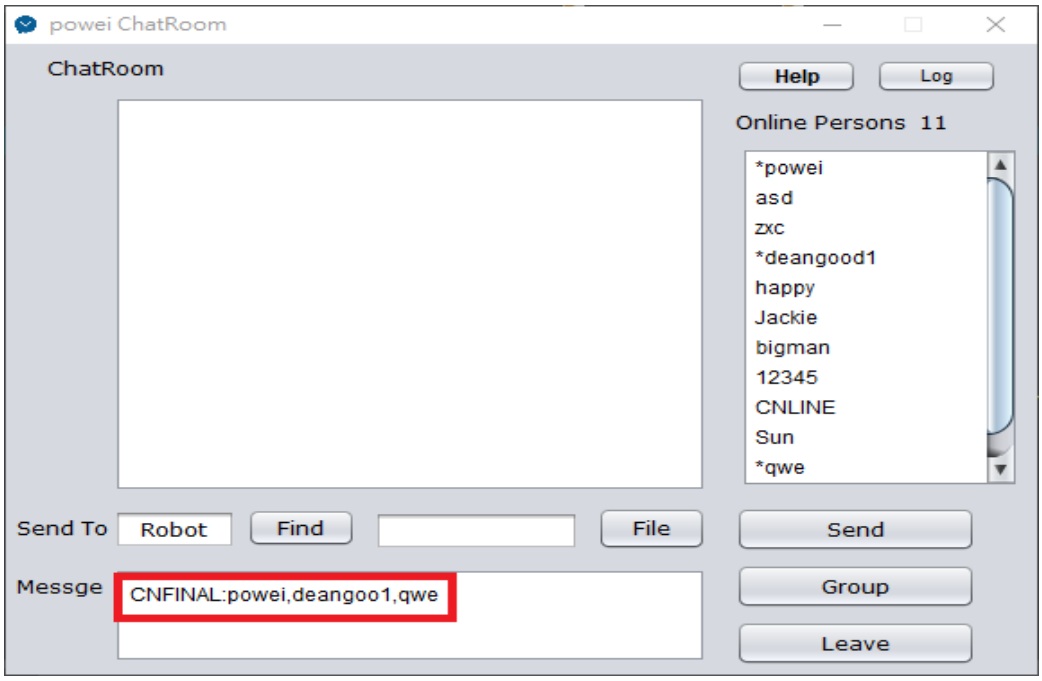


4. 接收檔案的使用者按下確認，及完成檔案的傳送和接收。



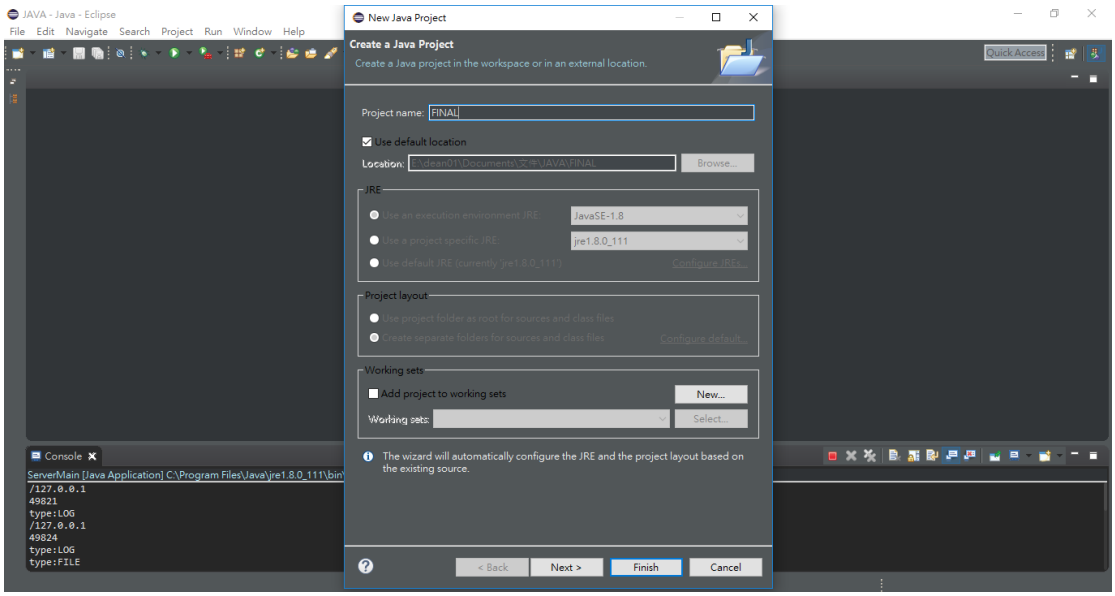
群組聊天:

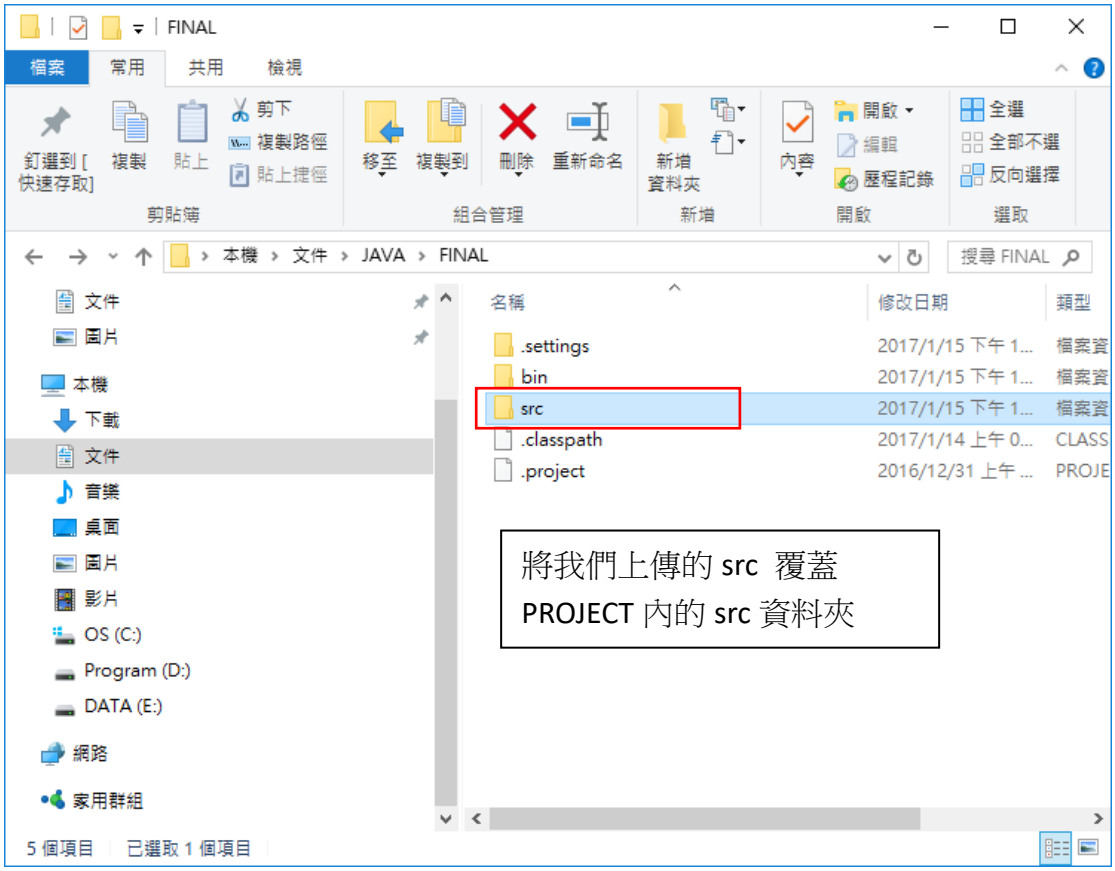
1. 使用者在 Message 的方塊中輸入:
   [群組名稱]: USER1, USER2,USER3…
   即可完成群組創建，並動於群組的成員發送訊息。
   群組內的每個成員都會收到傳訊的訊息，而群組外的使用者是無法收到訊息的，也無法傳送訊息給沒有加入的群組。



4.Instructions on how to run server & clients

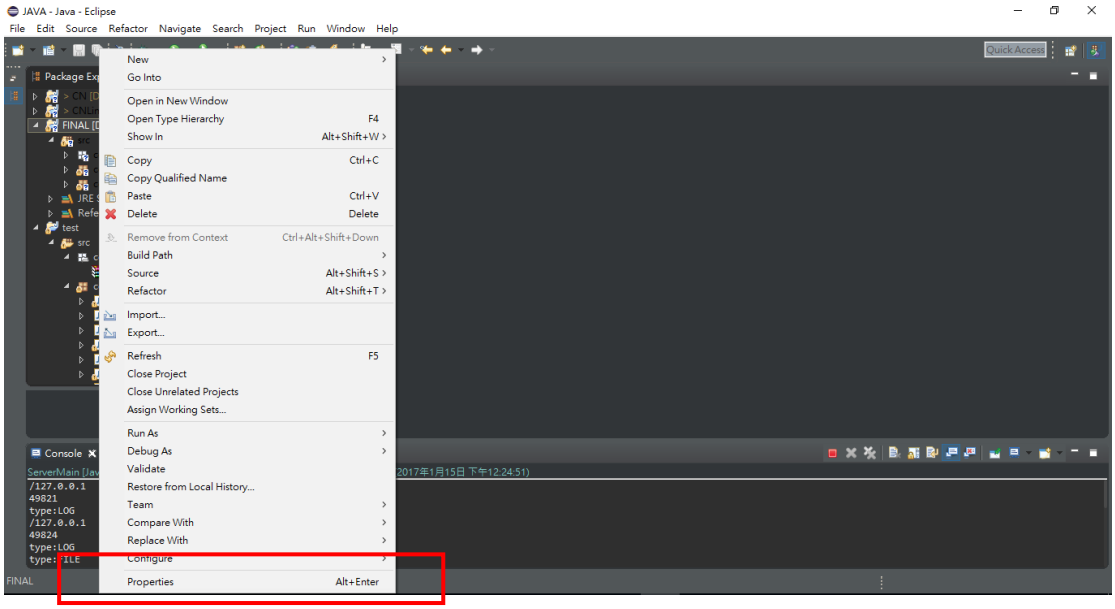
開 eclipse，新增一個 PROJECT，關閉 eclipse，到 PROJECT 的路徑，將我們的 src 覆蓋原本的 src 資料夾，把檔案全部 LOAD 進來

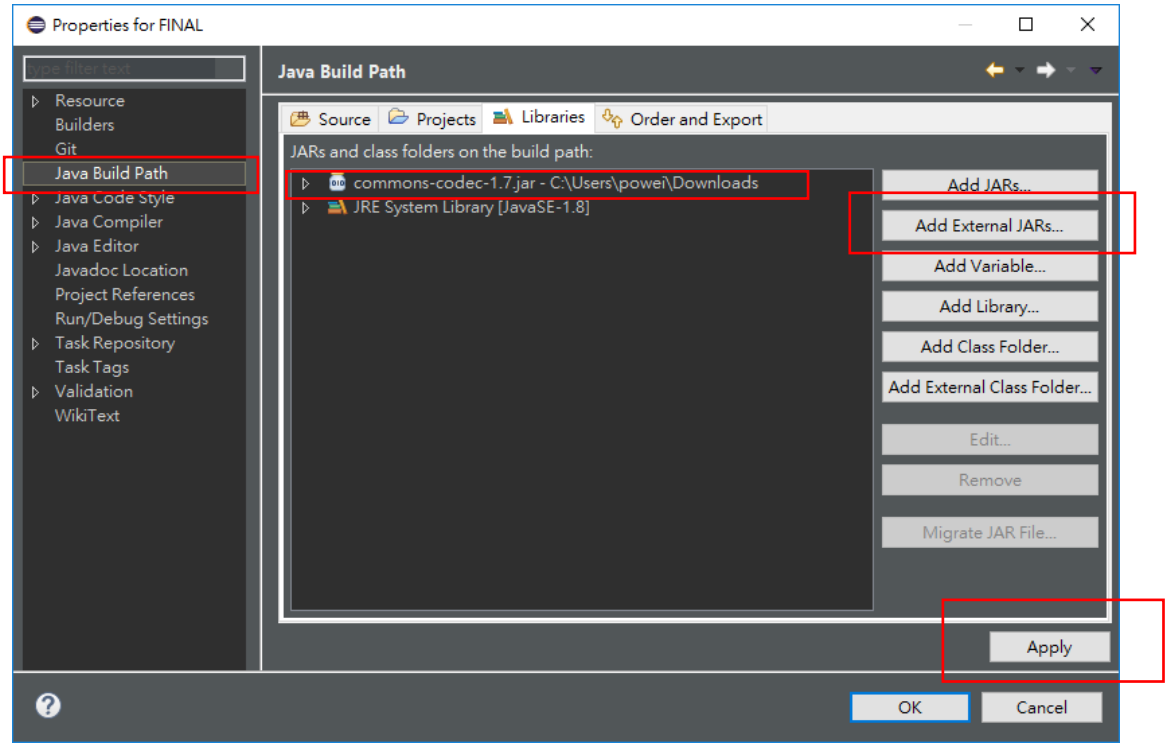




(2)因為 BONUS 的加密需要外部的演算法，必須做 LIBARAY LINKING

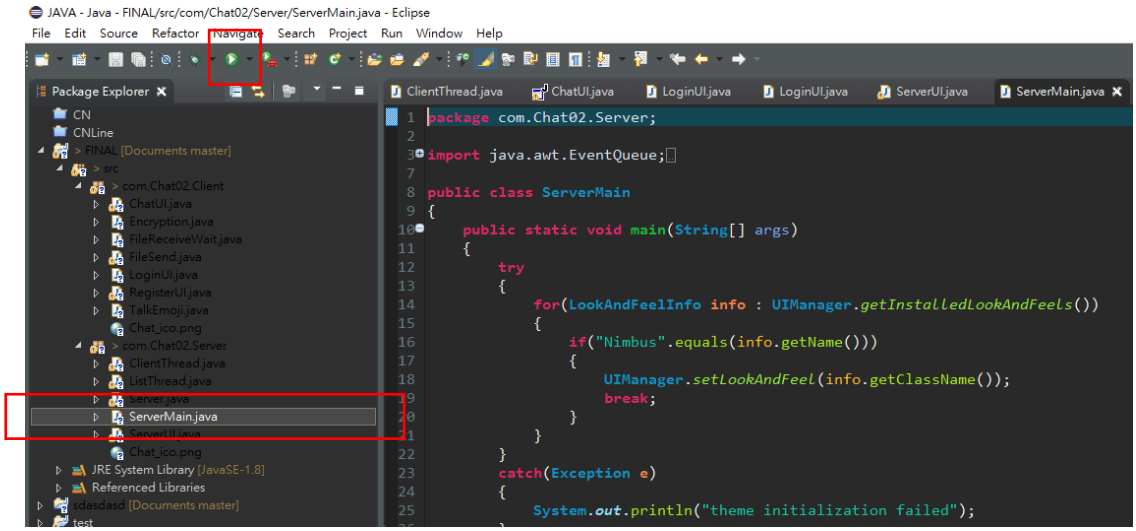
在專案右鍵>property>JAVA build path>Libaray> Add extereal jar>中加入我們附上的檔案 commoms-codec-1.7jar > 按下 Apply >然後將專案 refresh 即可



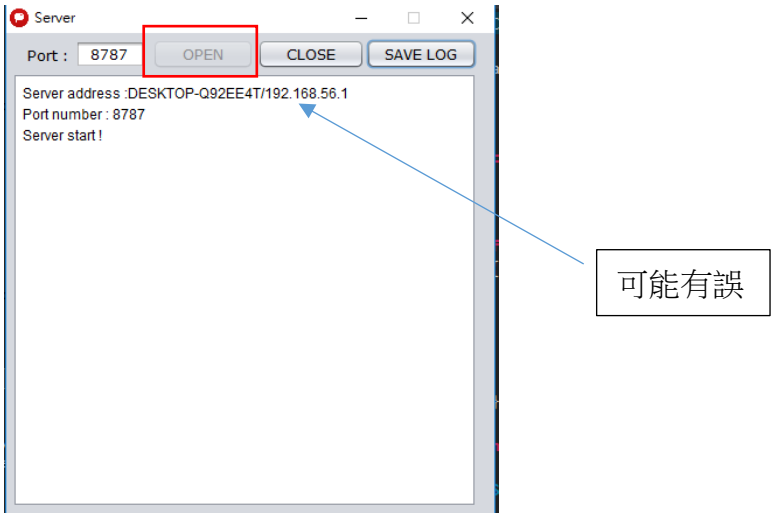


選取我們給的外部 JAR 檔案 > APPLY

(3)找到 server main 所在位置 ServerMain.java >build 即可啟動 Server



找的 client main 所在位置 ServerMain.java >build 即可啟動 Server



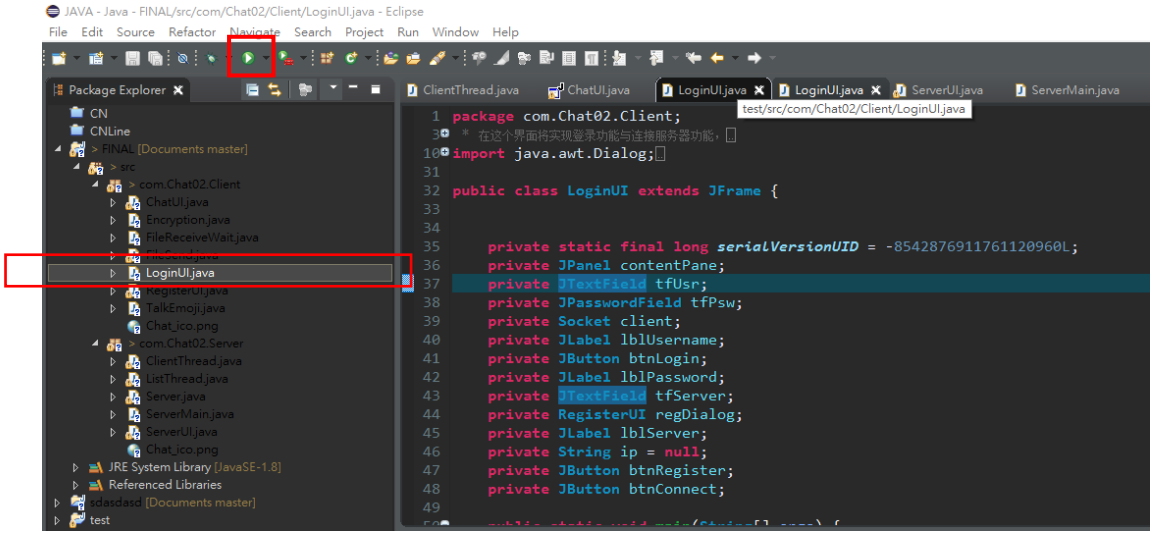
點選 OPEN 即可

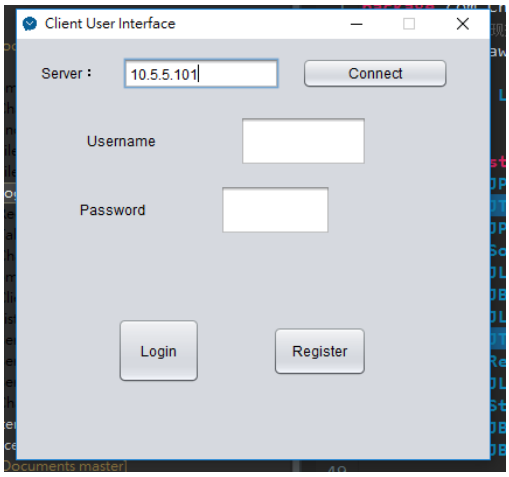
但是我們也有附上 build script for Linux

開啟兩個 terminal

分別輸入 bash buildServer.sh bash buildClient.sh 即可開啟 SERVER&CLIENT

(4.)找到 client main 所在位置 LoginUI.java >build 即可啟動 client





Server ip :打上目前 Server 的 IP 地址

由 CMD > ipconfig 即可知道 Server 目前的 public IP

Connect 後即可登入或註冊

5.System & program Design

1. TCP connection

For server:

Server 會一直等待 Client 來連接，每和一個 Client 建立連接就 new 一個 Thread。

```
while (!stop)
{
    Socket client = server.accept();// Blocking thread
    System.out.println(client.getInetAddress());
    System.out.println(client.getPort());
    Thread test = new Thread(new ClientThread
            (client, userMap,group, area), "ClientThread" + (i++));
    test.start();

}
```

For client:

一旦按下 Connect Button，Client 就會嘗試連接 textfield of server 中指定的 ip 位址，默認 connection port 是 8787，如果未連接成功，輸出錯誤資訊。

```
btnConnect.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String ip = tfServer.getText();//  "10.103.221.231"
        int port = 8787;
        try {
            client = new Socket(ip, port);
            LoginUI.this.ip = ip;
            setTitle(getTitle() + "...connect successfully");
            btnConnect.setEnabled(false);
        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            JDialog error = new JDialog(LoginUI.this, "Connection error", true);
            error.getContentPane().add(new Label("Server doesn't response! \r\n Pl
            error.setBounds(400, 200, 200, 100);
            error.setVisible(true);
            e1.printStackTrace();
        }
    }
});
```

2.Protocol Implementation

For Server：

Server 會依據 Client 發送的消息的第一個 header 判斷接下來的操作，是進行登錄驗證還是註冊還是發消息還是傳檔

```
while(true)
{
    //READ HEADER
    type = m_receiver.readLine();
    System.out.println("type:" + type);
    switch(type)
    {
    case "REG":
        register();
        break;
    case "LOG":
        login();
        break;
    case "MSG":
        getMessage();
        break;
    case "FILE":
        getRequest();
        break;
    case "GRP":
        System.out.println("GRP");
        createGroup();
        break;
    case "CMG":
        String cmg = m_receiver.readLine();
```

For Client：

以登錄為例，Client 會給 Server 發三行消息，LOG+username+password，根據 Server 返回的資訊判斷是否成功。Server 返回兩行：CSM（控制資訊，相當於 header）+Success/Failed/Repetition/Blank。

```
send.println("LOG");
send.println(name);
send.println(password);
String header = receive.readLine();
//String header = receive.readLine();//header = CSM
String data = receive.readLine();
switch (data) {
case "Success":
    System.out.println(header + " " + data);
    new ChatUI(client, name ,ip).setVisible(true);
    dispose();//close LoginUI
    break;
case "Failed":
    final Dialog faError = new Dialog(this, "Login Error", true);
    faError.add(new Label("    Username or Password is wrong. Plec
    faError.setBounds(600, 200, 190, 100);
    faError.setLayout(new FlowLayout());
    JButton faButton = new JButton("OK");
    faError.add(faButton);
    faButton.addActionListener(new ActionListener() {
    faError.setVisible(true);
    break;
case "Repetition":
```

File Transfer:

傳輸實作方式

由 A 向 server 請求 B 的 IP，然後竟由 P2P 的方式傳給 B

Multiple File transfer 的實現方式:

使用 Java FileChooser Class 的功能，可以產生一個視窗讓 Client 從電腦的儲存空間裡選擇多個檔案，然後將多個選擇的檔案儲存成 Array<file>的資料結構，一次一個檔案的傳輸方式將 Array 裡的檔案逐一傳給 Server，再由 Server 逐一將多個檔案傳給指定的 Client，完成多個檔案同時傳輸。

```
private void selectFile() {

    JFileChooser fcDlg = new JFileChooser();
      fcDlg.setDialogTitle("請選擇要讀取的檔案...");
      FileNameExtensionFilter filter = new
              FileNameExtensionFilter("JPEG file", "jpg", "jpeg");
      fcDlg.setMultiSelectionEnabled(true);
      fcDlg.showOpenDialog(frameMain);
      files = fcDlg.getSelectedFiles();
      textFieldSelect.setText(fcDlg.getCurrentDirectory()+files[0].getName());
      for(int i = 0; i < files.length; i++) {
          System.out.println(files[i].getName());
      }
```

傳輸進度顯示功能:

我們記錄已經傳輸的位元數目，將已傳輸資料和整份文件大小比例的結果以進度條的方式即時顯現給使用者，讓使用者可以了解整份檔案的傳輸進度、是否正常傳送。

```
for(int k = 0; k < files.length; k++) {
    System.out.println("file " +ipString);
    try {
        socket = new Socket(ipString, port);
    } catch (UnknownHostException e2) {
        // TODO Auto-generated catch block
        e2.printStackTrace();
    } catch (IOException e2) {
        // TODO Auto-generated catch block
        e2.printStackTrace();
    }
    System.out.println("In:" + files[k].getName() + " " + files[k].getPath());
    fileout = files[k];
    try {
        fileInputStream = new FileInputStream(fileout);
    } catch (FileNotFoundException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    bufferedInputStreamFile = new BufferedInputStream(
            fileInputStream);
    if (fileout != null) {
        try {
            // TODO
            fileLength = fileout.length();
            fileName = fileout.getName();
            inputStream = new DataInputStream(socket.getInputStream());
            bufferedOutputStreamNet = new BufferedOutputStream(
                    socket.getOutputStream());
            outputStream = new DataOutputStream(bufferedOutputStreamNet);
            int n = 0;
            int i = 0;
            int progress = 0;
```

4.Encryption

我們採用 AES 加密，在 Client 端指定 Key，在 Client 端加解密，防止在傳輸過程中被看到 Message 內容；

```
public static String encrypt(String data) throws Exception {
    String secret = "nihaoma";//
    MessageDigest dig = MessageDigest.getInstance(SEC_NORMALIZE_ALG);
    byte[] key = dig.digest(secret.getBytes(ENC));
    SecretKeySpec secKey = new SecretKeySpec(key, ALG);

    Cipher aesCipher = Cipher.getInstance(ALG);
    byte[] byteText = data.getBytes(ENC);

    aesCipher.init(Cipher.ENCRYPT_MODE, secKey);
    byte[] byteCipherText = aesCipher.doFinal(byteText);

    Base64 base64 = new Base64();
    return new String(base64.encode(byteCipherText), ENC);

}
```

```
public static String decrypt(String ciphertext) throws Exception {
    String secret = "nihaoma";// 
    MessageDigest dig = MessageDigest.getInstance(SEC_NORMALIZE_ALG);
    byte[] key = dig.digest(secret.getBytes(ENC));
    SecretKeySpec secKey = new SecretKeySpec(key, ALG);

    Cipher aesCipher = Cipher.getInstance(ALG);
    aesCipher.init(Cipher.DECRYPT_MODE, secKey);
    Base64 base64 = new Base64();
    byte[] cipherbytes = base64.decode(ciphertext.getBytes());
    byte[] bytePlainText = aesCipher.doFinal(cipherbytes);
    return new String(bytePlainText, ENC);
}
```

5.與機器人聊天

利用網路資源，提供的 API 介面，進行 HTTP post 和 get 操作

```
try {
    //String INFO = outArea.getText();
    String INFO = URLEncoder.encode(outArea.getText(), "utf-8");
    System.out.println("INFO:" + INFO);
    String getURL = "http://www.tuling123.com/openapi/api?key=" + APIKEY + "&info=" + INFO;
    System.out.println("URL:" + getURL);
    URL getUrl = new URL(getURL);
    HttpURLConnection connection = (HttpURLConnection) getUrl.openConnection();
    connection.connect();
    BufferedReader reader = new BufferedReader(new InputStreamReader( connection.getInputStream(), "utf-8"));
    StringBuffer sb = new StringBuffer();
    String line = "";
    while ((line = reader.readLine()) != null) {
        sb.append(line);
    }
    reader.close();
```

6.密碼強度判斷

利用 regular expression

```
public String checkPassword(String passwordStr) {
    String regexZ = "\\d*";
    String regexS = "[a-zA-Z]+";
    String regexT = "\\W+$";
    String regexZT = "\\D*";
    String regexST = "[\\d\\W]*";
    String regexZS = "\\w*";
    String regexZST = "[\\w\\W]*";

    if (passwordStr.matches(regexZ)) {
        return "weak";
    }
    if (passwordStr.matches(regexS)) {
        return "weak";
    }
    if (passwordStr.matches(regexT)) {
        return "weak";
    }
    if (passwordStr.matches(regexZT)) {
        return "medium";
    }
    if (passwordStr.matches(regexST)) {
        return "medium";
    }
    if (passwordStr.matches(regexZS)) {
        return "medium";
    }
    if (passwordStr.matches(regexZST)) {
        return "strong";
    }
    return passwordStr;

}
```