**Computer Networks Homework Two**

**- Java Socket Programming**

**程式主題:多人聊天室**

**組員: 101502542 林冠宇**

**101502557 劉冠廷**

**貢獻百分比:**

**林冠宇:Server端的製作、文件大綱與文件最後的整合-60%**

**劉冠廷:Client端的製作、參與部分文件編輯-40%**

**程式介紹：**

這是一個能夠使得多人同時連線聊天的的程式，是一個簡易的聊天室。透過開啟server端，我們稱其為NaServer，所有使用者都能透過執行NaClient的程式來達到連接NaServer的效果，並藉由NaClient的輸入訊息格，將訊息傳輸到NaServer，再經由NaServer推播給每位NaClient以達到多人聊天室的效果。

**程式分工以及實作動機：**

由於過往的課程以及自主練習，對於分工這件事我們都能夠比較了解該怎麼將一件工作做適當的拆解以及如何進行最後的整合動作。由於此次任務是有關於socket的應用，對於我們而言，我們第一個聯想到的便是曾經實作過的聊天室。但是對於那時候的我們而言，由於學習上對於程式還未有進一步的了解，所以我們僅僅只是實作出單純的一對ㄧ聊天，而且是透過command window進行運作，並未有GUI介面。所以此次經過討論，我們決定實作一個多人聊天室，且透過這個機會，學習如何操作java進行GUI的實作。

在經過討論後，決定：

林冠宇：進行所有有關於server端的實作，也就是實現NaServer的部分。

劉冠廷：進行所有有關於client端的實作，也就是實現NaClient的部分。

最後在一起進行整合，測試。

**組員參與程度：**

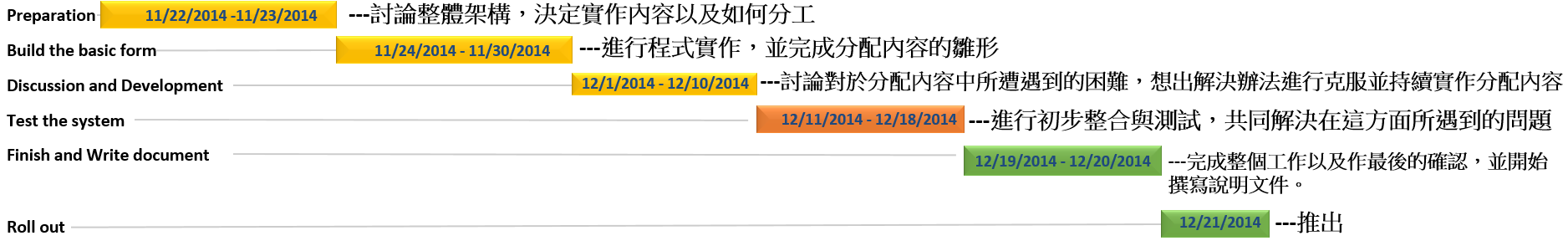
組員之間曾經合作過，且大家對於工作都有積極的參與，所以在參與度上都是全程參與討論以及實作的。

林冠宇：100% 劉冠廷：100%

**整體計劃流程與甘特圖：**

我們主要將整個工作粗略分為五大階段，也就是：

1. 討論整體架構，決定實作內容以及如何分工。
2. 進行程式實作，並完成分配內容的雛形。
3. 討論對於分配內容中所遭遇到的困難，想出解決辦法進行克服並持續實作分配內容。
4. 進行初步整合與測試，共同解決在這方面所遇到的問題。
5. 完成整個工作以及作最後的確認，並開始撰寫說明文件。



**程式安裝與使用說明:**

(1)需求環境:在任何擁有java編譯器以及具有安裝Java環境的電腦上都可以進行運作(編碼使用:UTF-8)

(2)使用說明:

<1>執行NaServer.java以開啟伺服端。

<2>執行NaClient.java，輸入基礎登入資訊。

<3>若要多人連線，只需多執行幾次NaClient.java即可，每執行一次即代表多一位使用者。

**參考網站:**

**1.http://docs.oracle.com/javase/tutorial/uiswing/components/panel.html**

**2.** [**http://zhidao.baidu.com/question/521817693.html**](http://zhidao.baidu.com/question/521817693.html)

**3.** **http://javapub.iteye.com/blog/683944**

**4.** **http://pydoing.blogspot.tw/2011/05/java-basic-concept-of-gui.html**

**流程圖:Server端簡易流程圖**

**NaClient**

**NaServer**

**紀錄訊息、IP**

**推播給每位使用者**

**輸入訊息**

**任一NaClient**

**流程圖:Client端簡易流程圖**

**server**

**登入介面**

**載入背景圖片**

**NaServer**

**使用者輸入名稱與ip位置**

**抓取系統時間**

**然後跟輸入資料一起傳給server**

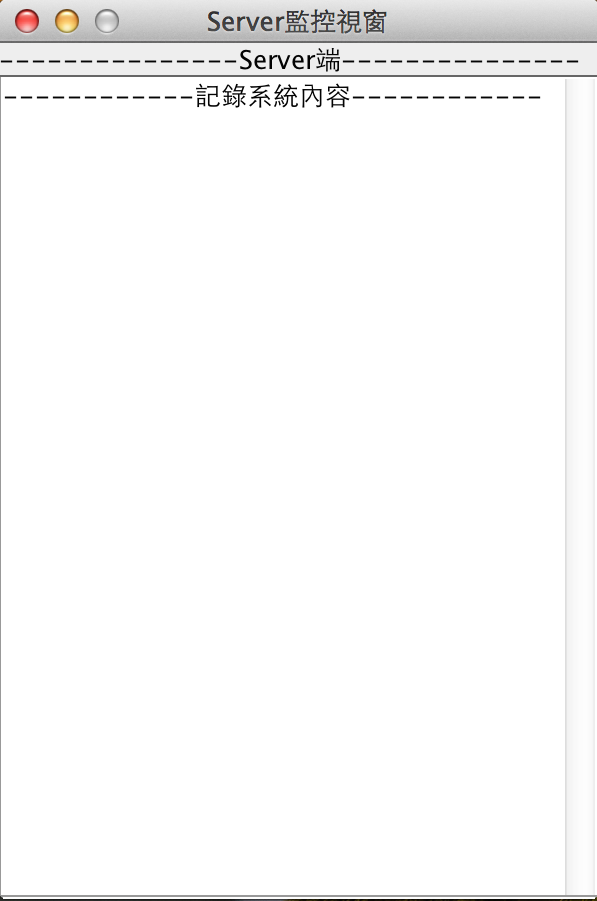
**聊天室主畫面**

**載入背景圖片**

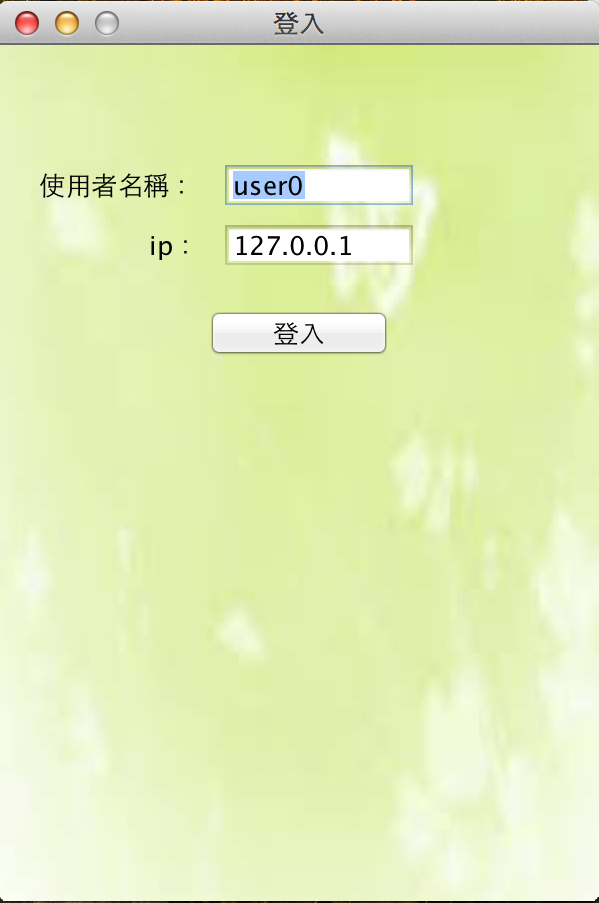
**程式執行截圖：**

1. 起始畫面：

**Server端：**

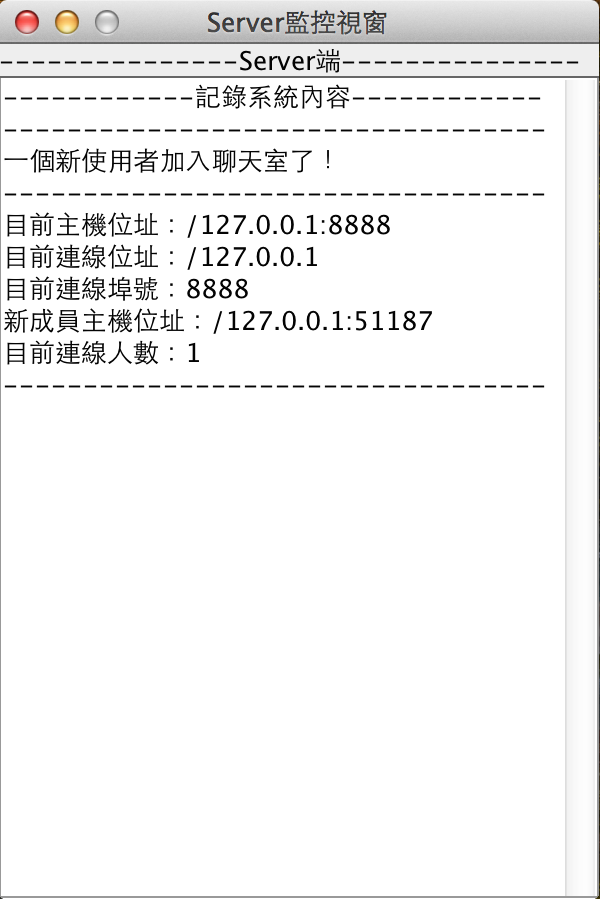
****

**Client端（登入畫面）：**

****

1. 登入之後的初始畫面：

Server端（有使用者加入之後）：



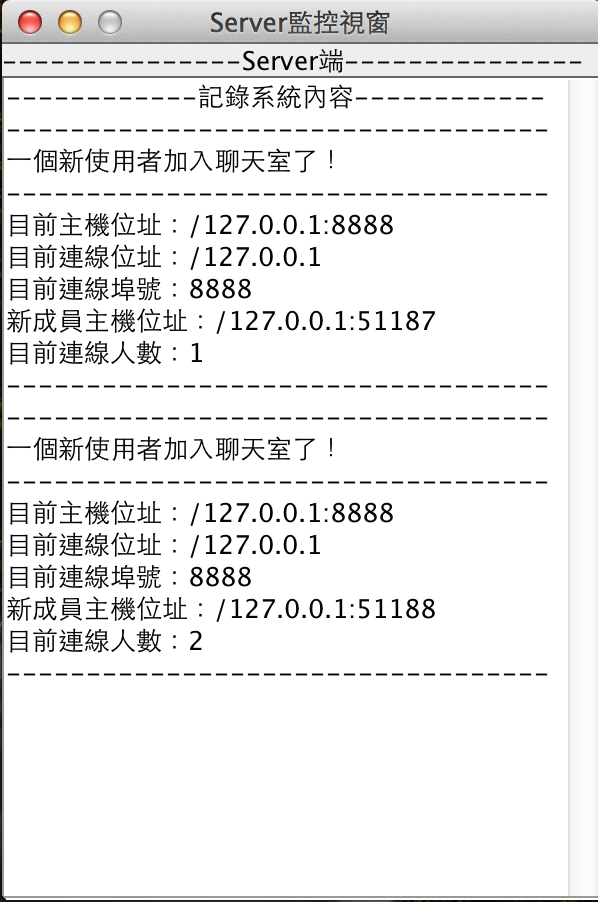
Client端(登入後的聊天室畫面)：



1. **第二位使用者加入**

**（多人連線,可以容納更多人）：**

Server端：

****

Client端（第一位使用者）：

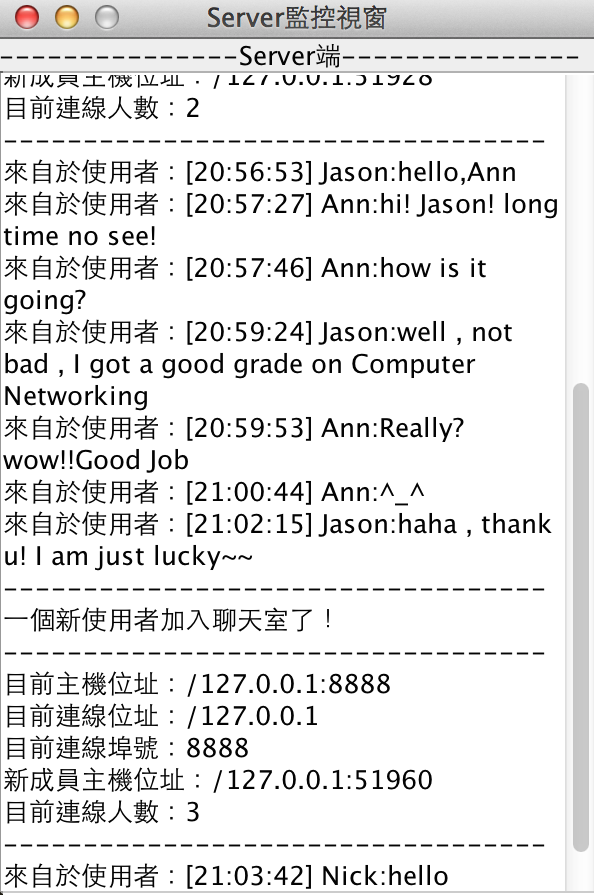
****

Client端（第二位使用者）：

****

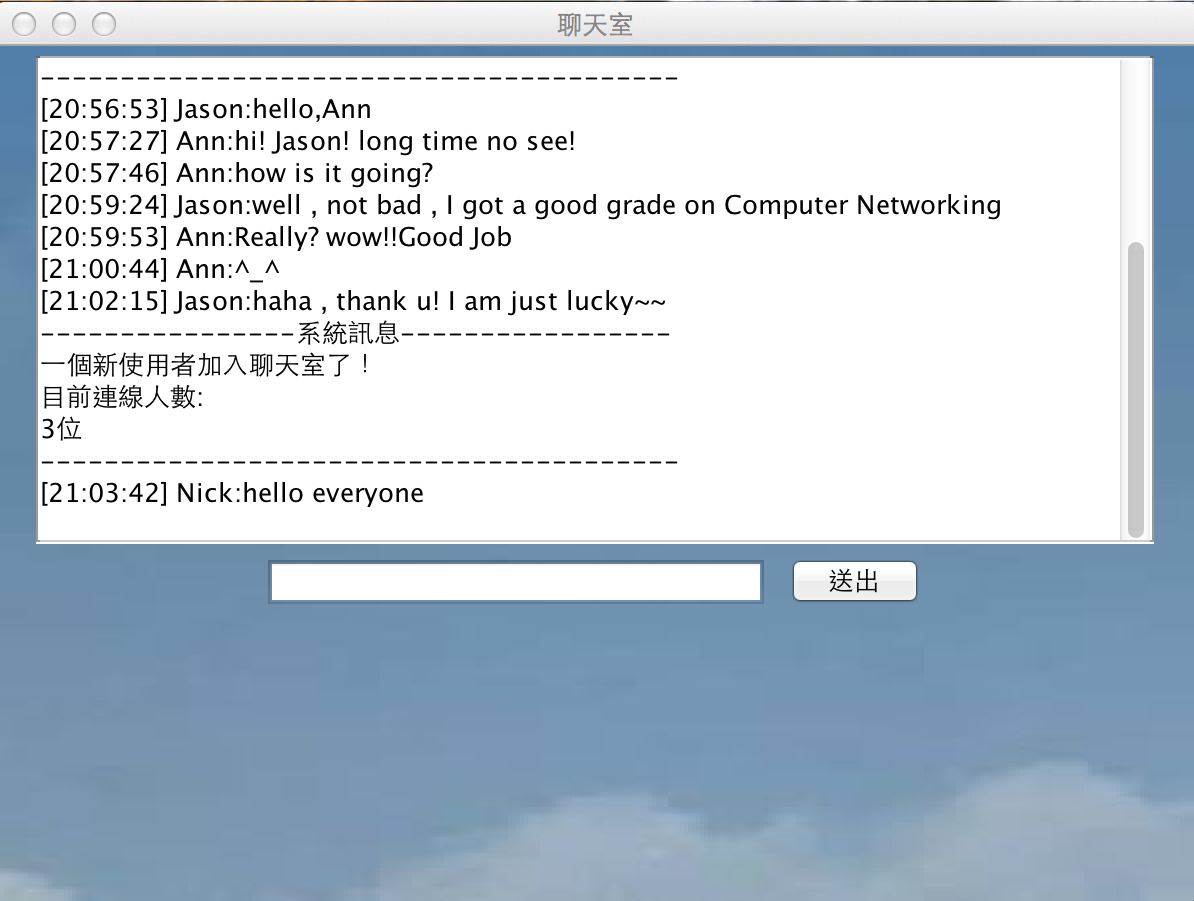
1. **開始聊天：**

**Server端：**

****

**Client端**

**（Jason）:**

****

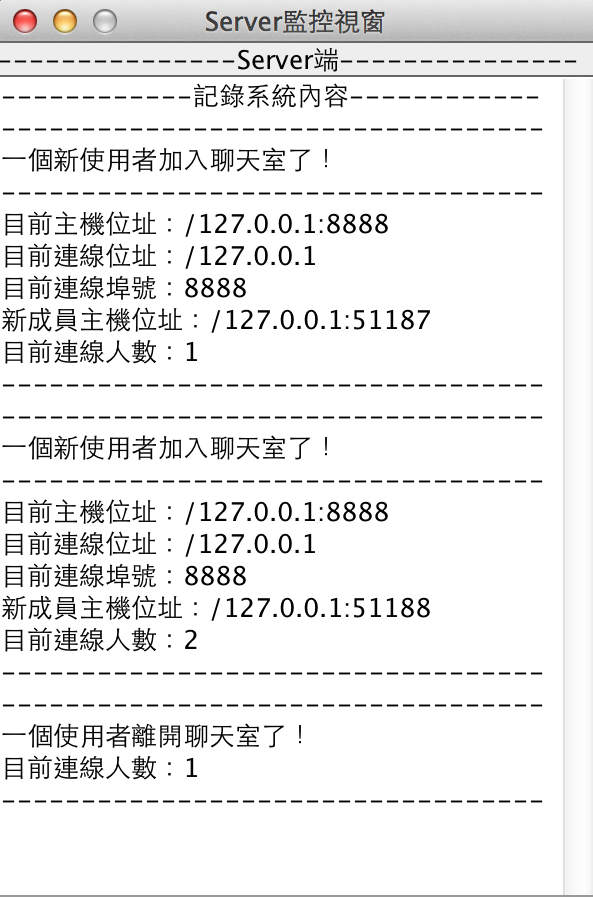
**(Ann):**

**(Nick):**

****

1. **使用者離開畫面：**

**Server端：**

****

**Client端：**

****

**心得與回饋：**

林冠宇：

能夠回到自己熟悉的語言，真的是一件很快樂的事情！

從我大一進來資工系開始，我第一個接觸到的語言便是java，也因此，我對於使用這個語言進行實作總是特別的開心！寫程式一直是一項很美好的事情，當然這是在沒有時間壓力的情況底下，我覺得這項工作能夠帶給你太多太多的樂趣以及成就感了。透過幾個關鍵字，幾句判斷式，就能夠實作出專屬於你自己獨一無二的東西不是很有趣嗎？不是很有成就感嗎？我到現在仍然記得第一次寫的Hello World，多少個程式設計師的起始都是從這裡開始的？至少對我來說，一句Hello World那背後所代表的意義，覺得不是將其印在command window上面這麼簡單，它帶給我的，是一個起始點，一個夢想的起始點。也因此，我超級熱愛實作，儘管學校課業壓力繁雜，常常讓人喘不過氣，但是我總要求自己能在閒暇時間多看一點，多學一點。一個人唯有像海綿那樣不斷地吸，那才有足夠的資本講出一些有建設性的話出來。而這一次的作業，又讓我學習了不少！

在許久沒碰這個語言的情況下，我重溫了執行序的開啟，重溫了class的應用，更重要的是，學習到了java的GUI的實作。什麼是layout？什麼是panel？這種東西我想大家都絕對不會太過陌生，但對於不同的語言，總會有不同的實作方法，對於使用Visual Studio裡的CLR主控台應用程式或是像matlab這種用拉的GUI之後，我有許久不曾碰到過用寫的GUI，這不僅是項挑戰，重點是，要怎麼把它跟socket裡的server以及client作結合？於是在與組員的討論之下，才有了這項成品。在一個團隊中，組員能夠進行溝通並互相討論問題癥結點是一項很重要的事情，而我也慶幸我能擁有這麼樣的一個隊友，我們一起克服了Scroller的使用；一起解決了執行序總是會炸掉的問題；一起嘗試了新的寫法；一起看著這個多人聊天室的誕生。縱使我們的聊天室很簡單很陽春，整份code以我的server舉例，一共也才200行出頭，但這中間我們遭遇到的挫折數量是很難估計的，但每經過了一個挫折，我們就又往前邁進了一點，我們就又成長了一點。

Java內部其實已經將socket包裝的非常好，太困難的東西我們都不需要去解決，我們要學的，是怎麼去使用，而很慶幸大一的時候陳國棟教授教會了我們這項應用，所以在這次的程式成品上，我們能夠從這個基礎慢慢往下延伸，進而達到了多人聊天室的效果。每次看到自己辛苦出來的成品總是很開心的，它就像自己親手栽培的一棵幼苗一樣，或許很瘦小，很不起眼，但它能夠帶給我們的成就感，外人著實很難體會，唯有曾經為它努力過，才能品嘗其中的滋味吧。感謝助教出了這麼一份作業，讓我能再次感受到java帶給我的感動！

劉冠廷:

剛開始寫的時候有種很陌生的感覺，因為已經蠻久沒寫過Java了。剛開始寫介面的時候感覺就像回到當初剛學java那樣，每種物件的使用方法都要再去熟悉一次，爾且有時候還會碰到一些問題。

在剛開始要寫連線部分時，有去看助教給的範例，但是把它放到eclipse上時卻發現沒有反應，當下找了許久還是沒有找到哪裡有問題，可能是自己環境問題也說不定。之後就開始看助教範例的code，也是看了許久雖然知道他是如何連線，但要實作出來還是有某個程度上的困難，剛說到的都是client的部分。然後就只好先去設計界面的部分，這部分我是先弄一個登入介面出來，讓使用者輸入基本資訊，然後再讓使用者進入主畫面，聊天室的主畫面是一個非常簡潔，也只有基本功能的聊天室介面。別小看這介面，他是我費了好多時間才給他實作出來的，雖然有點陽春，但是實作出來後也是有點成就感的。

在實作過程中，在做介面時還不了解畫布的用法，也費了滿久的時間。那時還不了解如何使用物件跟畫布，之後才慢慢知道如何把畫布跟物件做整合的動作。然後就是背景的部分，當初想加背景時就去網路上找方法，找到的方法百百種，但是能用的卻不多，在實作多種方法後才找出一種能用得最順手，比較不會出錯的方法，但是這方法卻要多出一個java檔。當初在了解這方法時也是不知道他在做啥，之後才知道他函式裡面在幹嘛，然後才能選擇我喜歡的背景加到聊天室中，我用了兩個畫布來加載兩種不同的背景，一個是登入畫面的，另一個是聊天室主畫面的，加完後有了背景感覺賞心悅目多了。另外就是有抓了時間在使用者輸入的時候，能看到雙方是啥時候輸出的，這樣有了前面幾項感覺起來整個架構就比較完整了。

在來就是連線的部分，我在宣告socket物件時不知道為啥要這樣做，他的原理不太了解，只知道照著這樣的步驟下去做就能實作出來。但是我了解到的是client端做輸出到server端跟接收server端的東西，而我都在接收的那部分上做文章，像抓時間就是在接收這邊做的。

最後就是覺得整個做完之後頗有成就感的，覺得自己在實作過程中coding的能力有得到改進。也大概了解了socket連線大致上是如何去做連線的動作，還有就是在解決問題上也能用比較有效率的方法去做解決。感覺起來比剛開始的時候，那時還不了解整個架構時，更了解了tcp這個東西。當初在課本上讀到的內容只有大概的概念，但是經過實作後，感覺上有對tcp有更深入的了解了。雖然過程中有許多困難，在解決上也費了相當多的時間。但是結束時卻有種海闊天空的感覺，就好像原來就是這樣，疑問得到解決之後放鬆的感覺非常舒服。

**程式內容介紹：**

1. **NaServer :**

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Iterator;

import java.util.Vector;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.ScrollPaneConstants;

public class NaServer extends JFrame implements ActionListener {

***//先定義執行序名字,之後才會分配值給它***

Thread primaryThread;

***//定義vector名字***

public static Vector systemMessage;

***//定義一個string***

public String localSocketAddress;

***//定義一個JLabel***

JLabel showWelcome = new JLabel("---------------Server端---------------");

***//顯示記錄區域***

JTextArea showtext = new JTextArea(15, 50);

***//創造主畫面,一個JFrame***

JFrame serverFrame = new JFrame("Server監控視窗");

***//取得serverControl的輸入***

String serverInput;

***//整個server的程式進入點***

public static void main(String args[]) {

new NaServer().setUpTheConnection();

}

public void setUpTheConnection() {

***//將元件加入主畫面***

serverFrame.add(showWelcome);

***//在主畫面中加入layout,以便加上scroller***

serverFrame.setLayout(new BorderLayout(300, 0));

***//當行的長度大於所分派的寬度時，將換行***

showtext.setLineWrap(true);

***//當行的長度大於所分派的寬度時，將在邊界（空白）處換行***

showtext.setWrapStyleWord(true);

***//設定為不可編輯的***

showtext.setEditable(false);

***//顯示初始文字***

showtext.append("------------記錄系統內容------------\n");

***//new出JScrollPane***

JScrollPane Scroller = new JScrollPane();

***//設定垂直滾動***

Scroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL\_SCROLLBAR \_ALWAYS);

***//設定水平滾動***

Scroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL\_SCROL LBAR\_NEVER);

Scroller.setViewportView(showtext);

***//增加滾動***

serverFrame.add(Scroller, BorderLayout.CENTER);

***//layout上方顯示文字***

serverFrame.add(showWelcome, BorderLayout.NORTH);

***//設定框架大小***

serverFrame.setSize(300, 450);

***//固定框架大小***

serverFrame.setResizable(false);

***//讓框架顯現***

serverFrame.setVisible(true);

***//儲存訊息用的vector***

systemMessage = new Vector();

try {

***//產生socket port***

ServerSocket serverSocket = new ServerSocket(8888);

while (true) {

***//等待是否有外部連線請求***

Socket socketSide = serverSocket.accept();

***//建立系統的input/output,此部分是用來取得socketSide的輸出資料***

PrintStream PSwriter = new PrintStream(

socketSide.getOutputStream());

***//將元件加入vector***

systemMessage.add(PSwriter);

***//使用執行緒,如此便可以實現多人連線***

***//此部分負責建立最主要的執行緒***

***//把connectionProcedure這個class new出來***

connectionProcedure connectionProcedure = new connectionProcedure(

socketSide);

***//將new出來的class分配進去執行序***

primaryThread = new Thread(connectionProcedure);

***//啟動主要執行緒***

primaryThread.start();

***//顯示訊息於server端***

showtext.append("----------------------------------\n");

showtext.append("一個新使用者加入聊天室了！\n");

showtext.append("----------------------------------\n");

showtext.append("目前主機位址：" + socketSide.getLocalSocketAddress()

+ ("\n") + "目前連線位址：" + socketSide.getInetAddress()

+ ("\n") + "目前連線埠號：" + socketSide.getLocalPort()

+ ("\n") + "新成員主機位址："

+ socketSide.getRemoteSocketAddress() + ("\n")

+ "目前連線人數：" + (primaryThread.activeCount() - 2)

+ ("\n"));

showtext.append("----------------------------------\n");

***//顯示訊息於client端***

***//先使用string來儲存想要顯示的訊息***

***//再利用擁有推播功能的function:toEveryOne來推播給每位client***

String line = "----------------系統訊息-----------------";

String welcomeMessage = "一個新使用者加入聊天室了！";

String onlineMessage = "目前連線人數:";

String onlinePeopleNum = Integer.toString(primaryThread

.activeCount() - 2) + "位";

String line2 = "----------------------------------------";

connectionProcedure.toEveryOne(line);

connectionProcedure.toEveryOne(welcomeMessage);

connectionProcedure.toEveryOne(onlineMessage);

connectionProcedure.toEveryOne(onlinePeopleNum);

connectionProcedure.toEveryOne(line2);

}

} catch (Exception ex) {

showtext.append("連線失敗！請檢查是否有正常開啟serverSide&clientSide");

}

}

***//此class負責連線之後的處理程序***

public class connectionProcedure implements Runnable {

***//此部分是用以暫存資料，採用bufferReader***

BufferedReader bufferedReader;

***//建立一個暫存socket***

Socket tempSocket;

***//此部分負責建立起接收的任務***

public connectionProcedure(Socket socketSide) {

try {

tempSocket = socketSide;

***//建立系統的input/output,此部分是用來取得socketSide的輸入資料***

InputStreamReader ISReader = new InputStreamReader(

tempSocket.getInputStream());

***//new出bufferReader***

bufferedReader = new BufferedReader(ISReader);

} catch (Exception ex) {

showtext.append("連線失敗！請檢查是否有正常開啟serverSide&clientSide");

}

}

***//此部分負責執行執行緒***

@Override

public void run() {

String Message;

try {

***//讀取使用者輸入資料***

while ((Message = bufferedReader.readLine()) != null) {

showtext.append("來自於使用者：" + Message + "\n");

toEveryOne(Message);

}

***//顯示系統訊息於server端介面***

showtext.append("----------------------------------\n");

showtext.append("一個使用者離開聊天室了！\n");

showtext.append("目前連線人數：" + (primaryThread.activeCount() - 3)

+ ("\n"));

showtext.append("----------------------------------\n");

***//顯示系統訊息於client端介面***

***//先使用string來儲存想要顯示的訊息***

***//再利用擁有推播功能的function:toEveryOne來推播給每位client***

String line = "----------------系統訊息-----------------";

String exitMessage = "一個使用者離開聊天室了！";

String onlineMessage = "目前連線人數:";

String onlinePeopleNum = Integer.toString(primaryThread

.activeCount() - 3) + "位";

String line2 = "----------------------------------------";

toEveryOne(line);

toEveryOne(exitMessage);

toEveryOne(onlineMessage);

toEveryOne(onlinePeopleNum);

toEveryOne(line2);

} catch (Exception ex) {

}

}

***//此部分用以告訴每位使用者聊天訊息，也就是進行推播的工作。***

public void toEveryOne(String Message) {

***//使用iterator來存取並輸出集合內的資料***

Iterator iterator = systemMessage.iterator();

***//判斷集合內部是否還有資料***

while (iterator.hasNext()) {

try {

***//取得集合內資料***

PrintStream writer = (PrintStream) iterator.next();

***//印出***

writer.println(Message);

***//重新更新緩衝區。***

writer.flush();

} catch (Exception ex) {

showtext.append("連線失敗！請檢查是否有正常開啟serverSide&clientSide");

}

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

//TODO Auto-generated method stub

}

}

**（2〉NaClient :**

import java.awt.BorderLayout;

import java.awt.Image;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.Socket;

import java.util.Calendar;

import java.util.Date;

import java.util.Locale;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.ScrollPaneConstants;

public class NaClient extends JFrame implements ActionListener{

private static final long serialVersionUID = 1L;

***//宣告圖片***

Image imgbackground;

Image imgBottom;

***//宣告背景畫布***

DrawablePanel mainPanel;

DrawablePanel panBottom;

***//宣告Socket變數***

Socket Mysock;

***//顯示區域***

JTextArea showtext = new JTextArea(15,45);

***//輸入區域***

JTextField input = new JTextField(20);

***//宣告畫布***

JPanel logPanel = new JPanel();

***//宣號兩個frame一個登入另一個是聊天室主畫面***

JFrame frame =new JFrame("登入");

JFrame frame2 =new JFrame("聊天室");

Toolkit toolkit = Toolkit.getDefaultToolkit();

***//宣告一些標籤與使用者輸入物件***

JLabel jname = new JLabel("使用者名稱：");

JLabel jip = new JLabel("ip：");

JTextField jtname = new JTextField("使用者1號",10);

JTextField jtip = new JTextField("127.0.0.1",10);

JLabel userstate = new JLabel("請輸入資料");

***//宣告使用者名子及位置ip***

String username;

String ip="";

***//宣告輸出及輸入變數***

BufferedReader Breader;

PrintStream Pwriter;

static Date date = new Date() ;

***//主程式開始***

public static void main(String[] args){

NaClient client = new NaClient();

}

***//設定登入宣告視窗物件***

NaClient (){

***//載入圖片***

LoadImage();

***//設定登入背景圖片***

panBottom = new DrawablePanel(imgBottom);

panBottom.setBounds(0, 0, 300, 450);

panBottom.setLayout(null);

***//建立視窗JFrame***

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(null);

frame.setSize(300,450);

frame.setResizable(false);

frame.getContentPane().add(panBottom);

frame.setVisible(true);

***//設定物件位置與大小***

jname.setBounds(20,60,80,20);

jtname.setBounds(110,60,100,20);

jip.setBounds(75,90,100,20);

jtip.setBounds(110,90,100,20);

***//加入到JPanel***

panBottom.add(jname);

panBottom.add(jtname);

panBottom.add(jip);

panBottom.add(jtip);

***//登入按鈕***

JButton logbutton = new JButton("登入");

***//按下設定監聽事件***

logbutton.addActionListener(this);

***//設定按鈕位置***

logbutton.setBounds(100,130,100,30);

***//加入畫布***

panBottom.add(logbutton);

***//離開***

addWindowListener(new WindowAdapter()

{

public void windowClosing(WindowEvent e){

System.out.println("離開聊天室");

System.exit(0);

}

});

}

//-------------------------背景

public void LoadImage() {

imgBottom = toolkit.getImage(NaClient.class

.getResource("login2.png"));

imgbackground = toolkit.getImage(NaClient.class

.getResource("login.png"));

}

***//設立連線***

private void MakeConnection(){

try{

***//請求建立連線***

Mysock = new Socket(ip,8888);

***//建立I/O資料流與取得輸入資料流***

InputStreamReader inReader = new InputStreamReader(Mysock.getInputStream());

***//放入暫存區***

Breader = new BufferedReader(inReader);

***//取得Socket的輸出資料流***

Pwriter = new PrintStream(Mysock.getOutputStream());

***//連線成功***

userstate.setText("網路建立-連線成功");

}catch(IOException ex ){

System.out.println("建立連線失敗");

}

}

***//設定聊天室視窗物件***

public void Myclient(){

***//載入圖片***

LoadImage();

mainPanel = new DrawablePanel(imgbackground);// 主畫面

mainPanel.setBounds(0, 0, 300, 300);

//JButton("送出")

JButton sButton = new JButton("送出");

***//設定按下監聽事件***

sButton.addActionListener(this);

***//對話區域-----***

***//設定換行條件與物件不可編輯***

showtext.setLineWrap(true);

showtext.setWrapStyleWord(true);

showtext.setEditable(false);

//JScrollPane

JScrollPane Scroller = new JScrollPane(showtext);

***//設定垂直滾動***

Scroller.setVerticalScrollBarPolicy(

ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS);

***//設定水平滾動***

Scroller.setHorizontalScrollBarPolicy(

ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_NEVER);

***//聊天室畫布加入該有物件***

mainPanel.add(Scroller);

mainPanel.add(input);

mainPanel.add(sButton);

***//設定聊天室frame物件***

frame2.getContentPane().add(mainPanel);

frame2.setSize(600,450);

frame2.setVisible(true);

frame2.setResizable(false);

***//離開***

addWindowListener(new WindowAdapter()

{

public void windowClosing(WindowEvent e){

System.out.println("離開聊天室");

System.exit(0);

}

});

}

***//按下按鈕之動作事件***

public void actionPerformed(ActionEvent e){

String str=e.getActionCommand();

***//如果按下的是登入按鈕***

if(str.equals("登入")){

***//設定名字***

username = jtname.getText();

***//設定ip，此程式目前無作用***

ip = jtip.getText();

***//狀態***

userstate.setText("設定"+username+":"+ip);

***//建立連線----***

MakeConnection();

***//建立接收資料執行緒----***

Thread readerTh = new Thread(new InReader());

readerTh.start();

frame.setVisible(false);

Myclient();

}else if(str.equals("送出")){

try{***//送出資料***

String gs = date.toString();

String[] AfterSplit = gs.split(" ");

Pwriter.println(("["+AfterSplit[3]+"] "+username+":"+input.getText()));

***//刷新該串流的緩衝。***

Pwriter.flush();

}catch(Exception ex ){

System.out.println("送出資料失敗");

}

***//清完輸入欄位***

input.setText("");

}

}

***//接收資料***

public class InReader implements Runnable{

public void run(){

String message;

try{

***//把接收到的資料顯示在showtext上***

while ((message = Breader.readLine()) != null){

showtext.append(message+'\n');

}

}catch(Exception ex ){ex.printStackTrace();}

}

}

}

**（3〉DrawPanel:**

import java.awt.Component;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.JPanel;

***//用jpanel設定背景圖片***

public class DrawablePanel extends JPanel

{

Image image1;

public DrawablePanel(Image image)

{

super();

image1 = image;

}

***//設定畫布範圍與要畫哪張圖***

protected void paintComponent(Graphics g) {

g.drawImage(image1, 0, 0, getWidth(), getHeight(), this);

}

}