電腦網路進階(chat room)

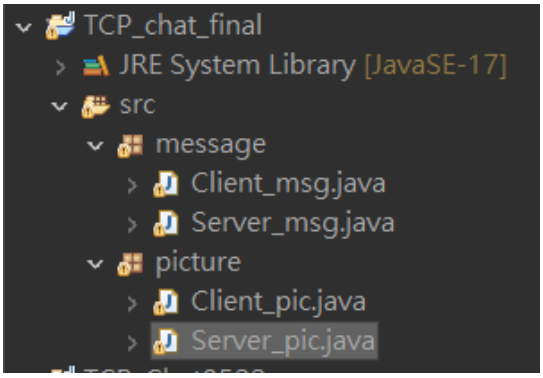
S0954043 郭柮杭
S0954052 范家豪

1. 配分

S0954043 郭柮杭(50%)-負責程式撰寫+書面報告
S0954052 范家豪(50%)-負責程式撰寫+書面報告

2. 程式碼(含程式碼註解)

礙於程式碼太多，放在 word 會影響可讀性及版面，而且毫無意義，因此只放傳訊息的 Server 端的程式碼截圖與我們所做的程式碼資料夾截圖。



傳訊息(Server)：

```
1 package message;
2 import java.awt.BorderLayout;
54
55 public class Server_msg {
56     private JFrame frame;
57     private JTextPane contentArea;
58     private JTextField txt_message;
59     private JTextField txt_port;
60     private JTextPane textPane;
61     private JButton btn_start;
62     private JButton btn_stop;
63     private JButton btn_send;
64     private JButton btn_image;
65     private JPanel northPanel;
66     private JPanel southPanel;
67     private JScrollPane centerPanel;
68     private ServerSocket serverSocket;
69     private ServerThread serverThread;
70     private ArrayList<ClientThread> clients;
71     private boolean isStart = false;
72     //主程式
73     public static void main(String[] args) {
74         new Server_msg();
75     }
76     //傳送訊息
77     public void send() throws BadLocationException, UnknownHostException {
78         if (!isStart) {
79             JOptionPane.showMessageDialog(frame, "伺服器還未啟動,不能傳送訊息!", "錯誤",JOptionPane.ERROR_MESSAGE);
80             return;
81         }
82         String message = txt_message.getText().trim();
83         if (message == null || message.equals("")) {
84             JOptionPane.showMessageDialog(frame, "不能為空!", "錯誤",JOptionPane.ERROR_MESSAGE);
85             return;
86         }
87         InetAddress loc = InetAddress.getLocalHost();
88         sendServerMessage(message);
89         StyledDocument document = (StyledDocument) contentArea.getDocument();
```

```
        document.insertString(document.getLength(), "Server:" +loc.getHostAddress()+">>"+txt_message.getText() + "\r\n", null);
        txt_message.setText(null);
    }
    public void sendServerMessage(String message) throws UnknownHostException {
        InetAddress loc = InetAddress.getLocalHost();
        for (int i = clients.size() - 1; i >= 0; i--) {
            clients.get(i).getWriter().println("Server:" +loc.getHostAddress()+">>"+ message);
            clients.get(i).getWriter().flush();
        }
    }

    //Server功能
    public Server_msg() {
        //視窗內的物件
        frame = new JFrame("Server");
        contentArea = new JTextPane();
        contentArea.setEditable(false);
        contentArea.setForeground(Color.blue);
        txt_message = new JTextField(45);
        txt_port = new JTextField("6666");
        btn_start = new JButton("啟動");
        btn_stop = new JButton("停止");
        btn_send = new JButton("傳送訊息");
        btn_image = new JButton("圖片");
        btn_stop.setEnabled(false);
        northPanel = new JPanel(new BorderLayout());
        northPanel.add(txt_message, "West");
        northPanel.add(btn_send, "Center");
        northPanel.add(btn_image, "East");
        northPanel.setBorder(new TitledBorder("訊息"));
        centerPanel = new JScrollPane(contentArea);
        southPanel = new JPanel();
        southPanel.add(new JLabel("Port"));
        southPanel.add(txt_port);
        southPanel.add(btn_start);
        southPanel.add(btn_stop);
        frame.setLayout(new BorderLayout());
        frame.add(northPanel, "North");
```

```
        frame.add(centerPanel, "Center");
        frame.add(southPanel, "South");
        frame.setSize(700, 400);
        frame.setVisible(true);
        //關閉視窗
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                if (isStart) {
                    closeServer();
                }
                System.exit(0);
            }
        });

        // 文字框按回車鍵時事件
        txt_message.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    send();
                } catch (BadLocationException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                } catch (UnknownHostException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
            }
        });

        // 單擊發送按鈕時事件
        btn_send.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                try {
                    send();
                } catch (BadLocationException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
```

```
            } catch (UnknownHostException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    });

    // 單擊啟動伺服器按鈕時事件
    btn_start.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            int port;
            try {
                try {
                    port = Integer.parseInt(txt_port.getText());
                } catch (Exception e1) {
                    throw new Exception("埠號為正整數!");
                }
                serverStart(port);
                StyledDocument document = (StyledDocument) contentArea.getDocument();
                document.insertString(document.getLength(), "Server已成功啟動!! " + "Port: " + port + "\r\n" , null);
                //contentArea.setText("Server已成功啟動!! " + "Port: " + port + "\r\n");
                JOptionPane.showMessageDialog(frame, "伺服器成功啟動!");
                btn_start.setEnabled(false);
                txt_port.setEnabled(false);
                btn_stop.setEnabled(true);
            }
            catch (Exception exc) {
                JOptionPane.showMessageDialog(frame, exc.getMessage(),"錯誤", JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    // 單擊停止伺服器按鈕時事件
    btn_stop.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            try {
                closeServer();
                btn_start.setEnabled(true);
```

```
                    txt_port.setEnabled(true);
                    btn_stop.setEnabled(false);
                    contentArea.setText("伺服器成功停止!\r\n");
                    JOptionPane.showMessageDialog(frame, "伺服器成功停止!");
                }
                catch (Exception exc) {
                    JOptionPane.showMessageDialog(frame, "停止伺服器發生異常!", "錯誤",
                            JOptionPane.ERROR_MESSAGE);
                }
            }
        });
    }

    // 啟動伺服器
    public void serverStart(int port) throws IOException {
        try {
            clients = new ArrayList<ClientThread>();
            serverSocket = new ServerSocket(port);
            serverThread = new ServerThread(serverSocket, 2);
            serverThread.start();
            isStart = true;
        }
        catch (BindException e) {
            isStart = false;
            throw new BindException("啟動伺服器異常!");
        }
    }

    // 關閉伺服器
    @SuppressWarnings("deprecation")
    public void closeServer() {
        try {
            if (serverThread != null)
                serverThread.stop();// 停止伺服器執行緒

            for (int i = clients.size() - 1; i >= 0; i--) {
                // 給所有線上使用者傳送關閉命令
                clients.get(i).getWriter().println("CLOSE");
```

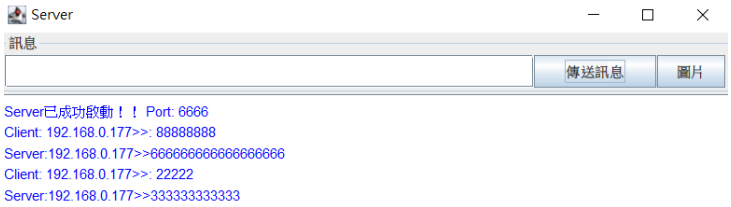
```
                clients.get(i).getWriter().flush();
                // 釋放資源
                clients.get(i).stop();// 停止此條為客戶端服務的執行緒
                clients.get(i).reader.close();
                clients.get(i).writer.close();
                clients.get(i).socket.close();
                clients.remove(i);
            }
            if (serverSocket != null) {
                serverSocket.close();// 關閉伺服器端連線
            }
            isStart = false;
        } catch (IOException e) {
            e.printStackTrace();
            isStart = true;
        }
    }


    // 伺服器執行緒
    class ServerThread extends Thread {
        private ServerSocket serverSocket;
        private int max;// 人數上限

        // 伺服器執行緒的構造方法
        public ServerThread(ServerSocket serverSocket, int max) {
            this.serverSocket = serverSocket;
            this.max = max;
        }

        public void run() {
            while (true) {// 不停的等待客戶端的連結
                try {
                    Socket socket = serverSocket.accept();
                    ClientThread client = new ClientThread(socket);
                    client.start();// 開啟對此客戶端服務的執行緒
                    clients.add(client);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    // 為一個客戶端服務的執行緒
    class ClientThread extends Thread {
        private Socket socket;
        private BufferedReader reader;
        private PrintWriter writer;
        //private User user;

        public BufferedReader getReader() {
            return reader;
        }

        public PrintWriter getWriter() {
            return writer;
        }
/*
        public User getUser() {
            //return user;
        }
*/
        // 客戶端執行緒的構造方法
        public ClientThread(Socket socket) {
            try {
                this.socket = socket;
                reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                writer = new PrintWriter(socket.getOutputStream());
                // 接收客戶端的基本使用者資訊
                String inf = reader.readLine();
                writer.flush();

            } catch (IOException e) {
                e.printStackTrace();
```

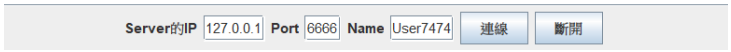
```
307                 this.socket = socket;
308                 reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
309                 writer = new PrintWriter(socket.getOutputStream());
310                 // 接收客戶端的基本使用者資訊
311                 String inf = reader.readLine();
312                 writer.flush();
313 
314             } catch (IOException e) {
315                 e.printStackTrace();
316             }
317         }
318 
319         @SuppressWarnings("deprecation")
320         public void run() {// 不斷接收客戶端的訊息,進行處理。
321             String message = null;
322             while (true) {
323                 try {
324                     message = reader.readLine();// 接收客戶端訊息
325 
326                     if (message.equals("CLOSE"))// 下線命令
327                     {
328                         // 斷開連線釋放資源
329                         reader.close();
330                         writer.close();
331                         socket.close();
332 
333                         // 向所有線上使用者傳送該使用者的下線命令
334                         for (int i = clients.size() - 1; i >= 0; i--) {
335                             //clients.get(i).getWriter().println(
336                                     //"DELETE@" + user.getName());
337                             clients.get(i).getWriter().flush();
338                         }
339                     }
340                     else {
341                         dispatcherMessage(message);// 轉發訊息
342                     }
343                 } catch (IOException | BadLocationException e) {
344                     e.printStackTrace();
```

```
331                         socket.close();
332 
333                         // 向所有線上使用者傳送該使用者的下線命令
334                         for (int i = clients.size() - 1; i >= 0; i--) {
335                             //clients.get(i).getWriter().println(
336                                     //"DELETE@" + user.getName());
337                             clients.get(i).getWriter().flush();
338                         }
339                     }
340                     else {
341                         dispatcherMessage(message);// 轉發訊息
342                     }
343                 } catch (IOException | BadLocationException e) {
344                     e.printStackTrace();
345                 }
346             }
347         }
348 
349         // 轉發訊息
350         public void dispatcherMessage(String message) throws BadLocationException {
351             StringTokenizer stringTokenizer = new StringTokenizer(message, "@");
352             String source = stringTokenizer.nextToken();
353             String owner = stringTokenizer.nextToken();
354             String content = stringTokenizer.nextToken();
355             message = source + ": " + content;
356             StyledDocument document = (StyledDocument) contentArea.getDocument();
357             document.insertString(document.getLength(), message + "\r\n" , null);
358             //contentArea.setText(message + "\r\n");
359             if (owner.equals("ALL")) {// 群發
360                 for (int i = clients.size() - 1; i >= 0; i--) {
361                     clients.get(i).getWriter().println(message);
362                     clients.get(i).getWriter().flush();
363                 }
364             }
365         }
366     }
367 }
368 
```

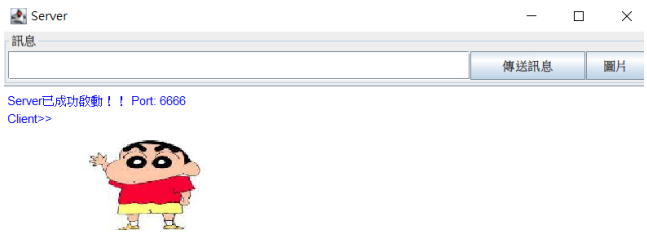
3. 程式結果截圖

傳訊息：





傳圖片：

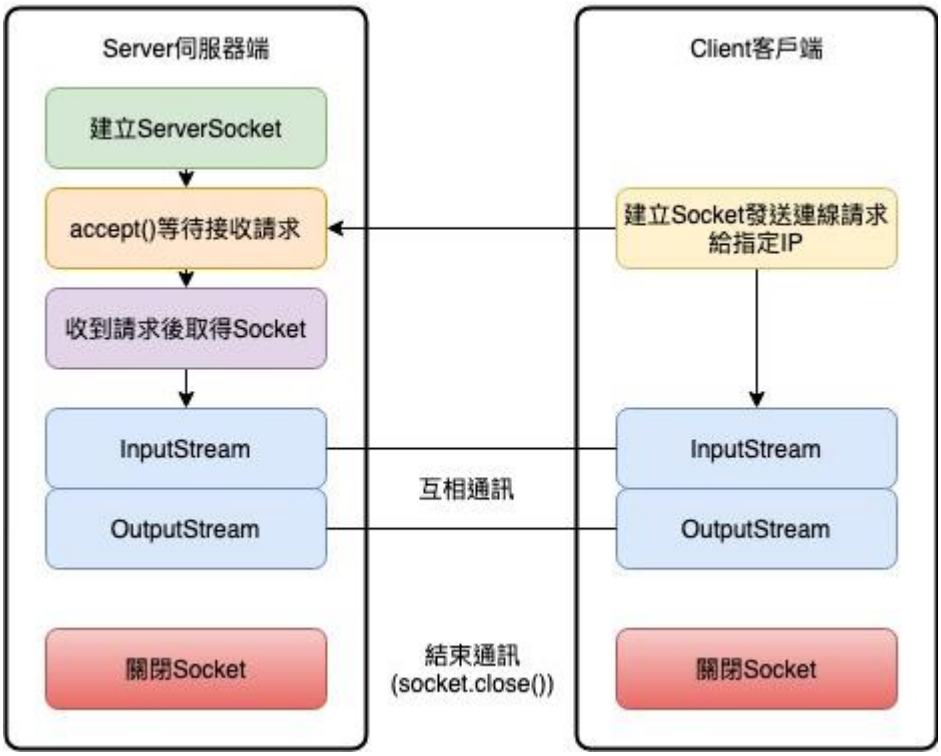


4. 程式功能說明

使用 TCP Socket 實作聊天室功能，能傳送及接收訊息或圖片，並將訊息及圖片顯示於畫面上。

5. 程式流程說明

開啟 server 端和 client 端，於 client 端輸入 server IP 和 port 進行連線，成功連線後，雙方可輸入訊息，按下傳送按鈕傳送到另一端，並於兩端畫面顯示，或是按下圖片按鈕可選取圖片，使用 TCP 傳送至另一端，並於兩端畫面上顯示，於 client 端按下結束按鈕則結束程式。



6. 心得

這次花了許多時間在學習如何撰寫將字串從 client 端傳給 server 端、server 端傳給 client 端等程式，而且撰寫期間由於雙方的程式撰寫風格不同，導致在 debug 時會花更多時間去理解對方的程式，以下是我們在實作過程中有遇到的問題：1.**圖片無法傳過去**。傳過去有可能是亂碼或根本顯示不出來，甚至是跳出錯誤訊息。2. 整**合起來會有問題**。單獨傳訊息或圖片都沒有問題，但是只要一整合在一起就會跑出一大堆問題，像是只會顯示圖片或只會顯示訊息，或是視窗直接卡死。3. **無法重複傳送**。圖片無法傳第二次，只要傳第二次，就會跳出 Socket closed，即使知道問題出在哪裡仍無法順利解決，試了很多種辦法都沒辦法在 socket 不關閉的情況下成功傳送圖片。4. **對於撰寫 Java 視窗程式語法尚未熟悉**。即使之前其中一位組員學過 Java 的視窗程式，但仍然在 UI

介面上卡了很久，有時候試了很久才發現是視窗程式漏寫某些東西，而不是傳輸的地方寫錯。此次作業是將課堂所學的理論知識應用到實務方面，實際操作才更深刻體會到原來創建一個雙方可以順暢溝通的環境是需要花費多少心力。