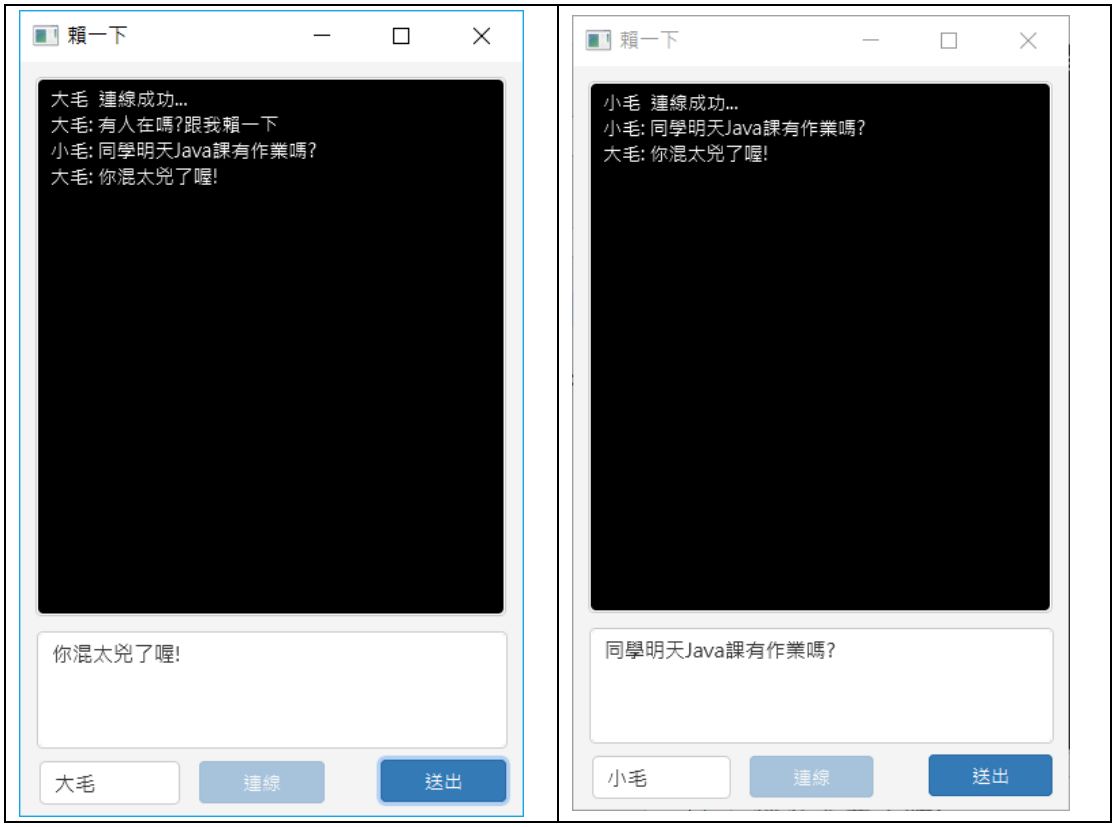
我也會 Line 喔!

網路(Network)應用程式:賴一下!

只要在網路傳送訊息、檔案，都會用到 Java 的 net 套件(FTP, P2P,網路通訊 …)。



這裡介紹的網路應用程式—多人聊天室，不是很簡單，你需要具備進階 Java 程式的能力：

- GUI 設計
- 自訂事件方法
- 多執行緒
- 例外處理
- 網路 Socket
- ArrayList< >或 HashSet< > 進階資料結構的使用

基本練功:server, client

Server 端: ServerSocket
ServerSocket.listen()監聽是否有連線進來
ServerSocket.accept()取得 client 端連線的 Socket 物件

Client 端: Socket

當 Server 和 Client 連接後:
準備兩個水管：
in 進來的
out 出去的

接收進來的對方訊息:
in.readLine()

送出訊息給對方
out.println()

伺服器

```
package chapter31_chatting_tutorial;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class MyServer {
	public static void main(String[] args) throws IOException {

		//伺服器初始化
		ServerSocket server = new ServerSocket(1024);
```

```
            //等待連線...
            System.out.println("等待連線...");
            Socket socket = server.accept();

            //建立串流
            DataInputStream fromClient = new DataInputStream(socket.getInputStream());
            DataOutputStream toClient = new DataOutputStream(socket.getOutputStream());

            //讀訊息  伺服器 read 讀取次數必須與 client 配合
            //讀不到資料時會天荒地老等 read 下去 甚至會打死結
            String msg = fromClient.readUTF();

            //送出訊息
            String response = String.format("伺服器回應:%s", msg);
            toClient.writeUTF(response);

            System.out.println(msg);
        }
}
```

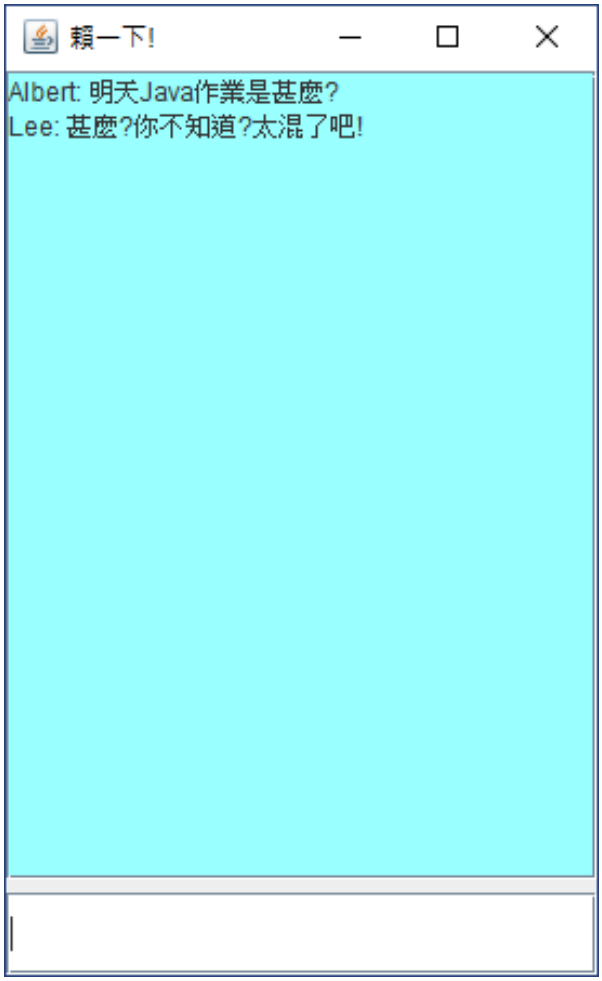
Client 端

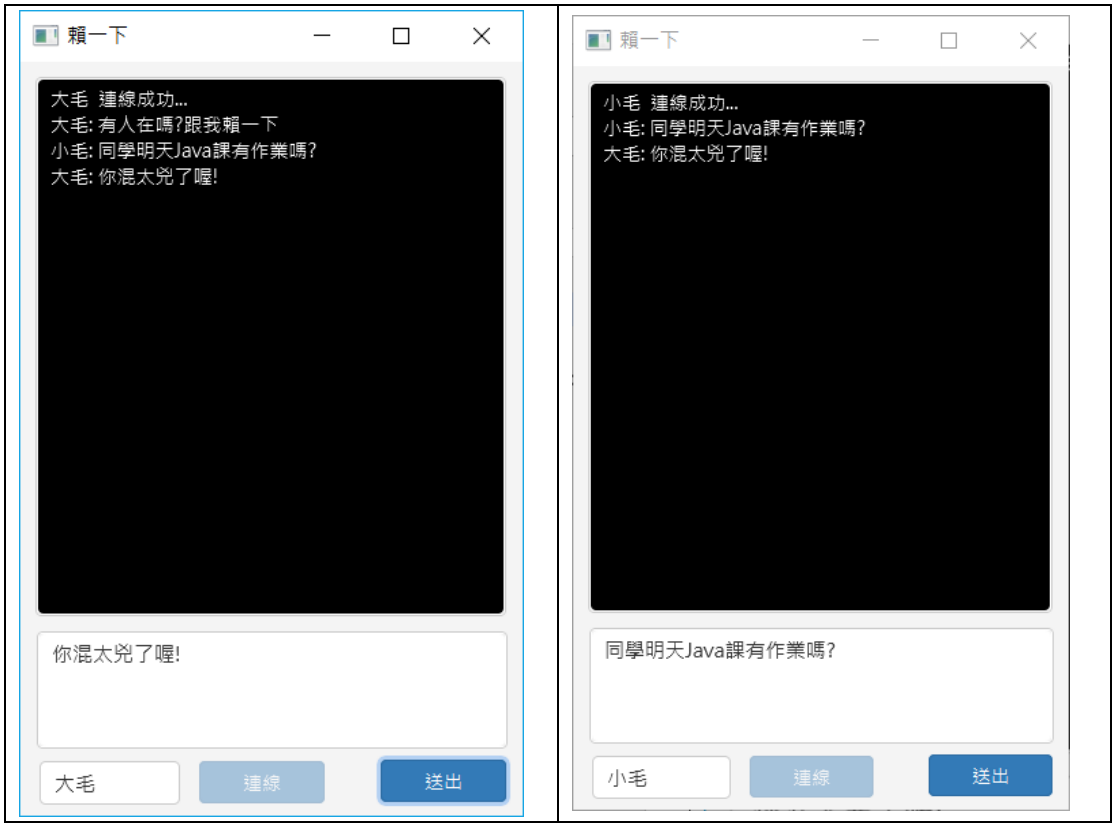
```
package chapter31_chatting_tutorial;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;

public class MyClient {

    public static void main(String[] args) throws IOException {
            Socket socket = new Socket("localhost", 1024);
            DataInputStream fromServer = new DataInputStream(socket.getInputStream());
```

```
            DataOutputStream toServer = new DataOutputStream(socket.getOutputStream());

            //送出訊息
            toServer.writeUTF("李大同");

            //讀入
            System.out.println(fromServer.readUTF());
        }
}
```

完成一個簡單的聊天室程式。

會寫這個程式可以去 Line 工作嗎?


賴一下!
Albert: 明天Java作業是甚麼?
Lee: 甚麼?你不知道?太混了吧!



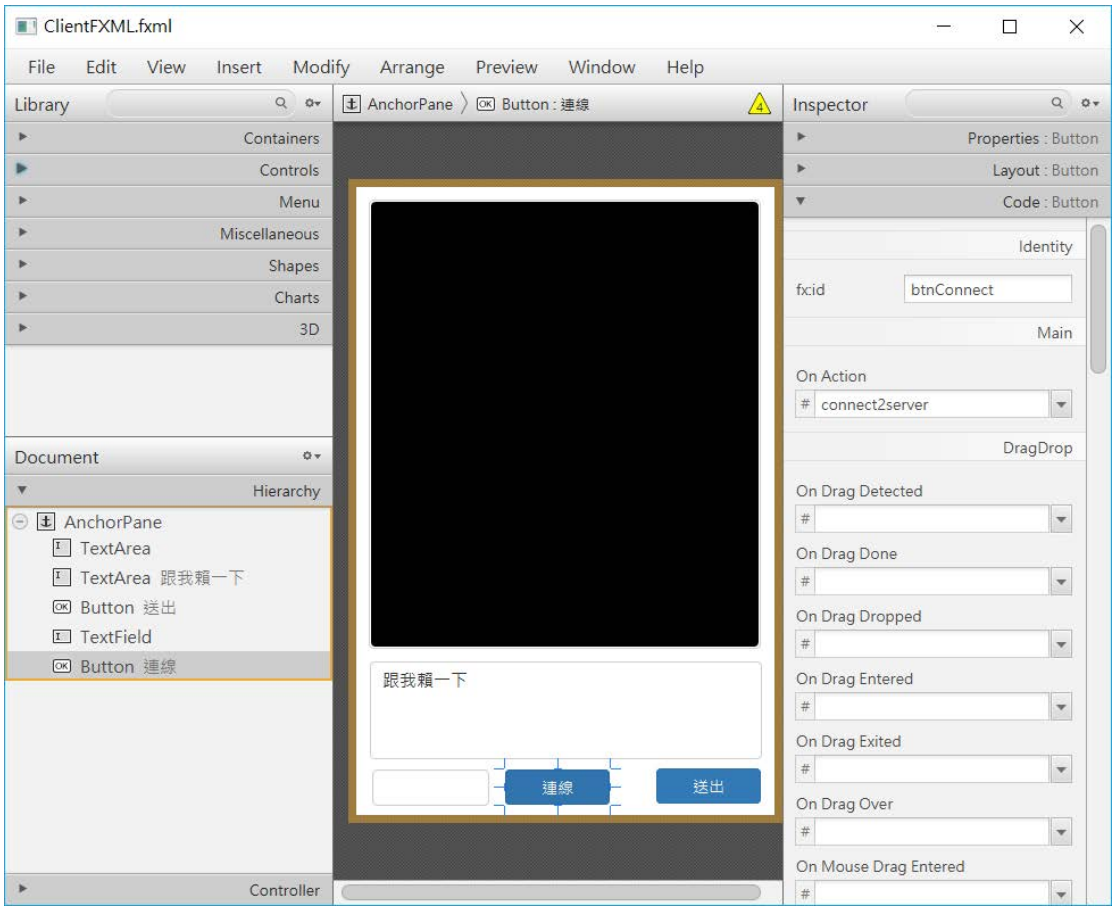
Client 端

ClientFXMLMain.java

```
package chapter31_chatting_tutorial;

import java.io.IOException;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class ClientFXMLMain extends Application {
```

```
    @Override
    public void start(Stage primaryStage) throws IOException {

        Parent root = FXMLLoader.load(this.getClass().getResource("ClientFXML.fxml"));

        Scene scene = new Scene(root);

        primaryStage.setTitle("賴一下");
        primaryStage.setScene(scene);
        primaryStage.show();
        primaryStage.setOnCloseRequest(e -> {
            Platform.exit();
            System.exit(0);
        });
    }

    public static void main(String[] args) {
        launch(args);
    }

}
```

ClientFXML.fxml




```
<?xml version="1.0" encoding="UTF-8"?>


<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>


<AnchorPane id="AnchorPane" prefHeight="552.0" prefWidth="368.0" stylesheets="@bootstrap3.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="chapter31_chatting_tutorial.ClientFXMLController">
    <children>
        <TextArea fx:id="display" editable="false" layoutX="11.0" layoutY="11.0" prefHeight="395.0" prefWidth="345.0" stylesheets="@myTextArea.css" wrapText="true" />
        <TextArea fx:id="input" layoutX="12.0" layoutY="417.0" prefHeight="86.0" prefWidth="345.0" text="跟我賴一下" wrapText="true" />
        <Button fx:id="btnSubmit" layoutX="264.0" layoutY="511.0" mnemonicParsing="false" onAction="#send" prefHeight="45.0" prefWidth="92.0" styleClass="primary" text="送出" />
        <TextField fx:id="textfield_name" layoutX="14.0" layoutY="512.0" prefHeight="32.0" prefWidth="103.0" />
        <Button fx:id="btnConnect" layoutX="131.0" layoutY="512.0" mnemonicParsing="false" onAction="#connect2server" prefHeight="45.0" prefWidth="92.0" styleClass="primary" text="連線" />
    </children>
```

```
</AnchorPane>
```

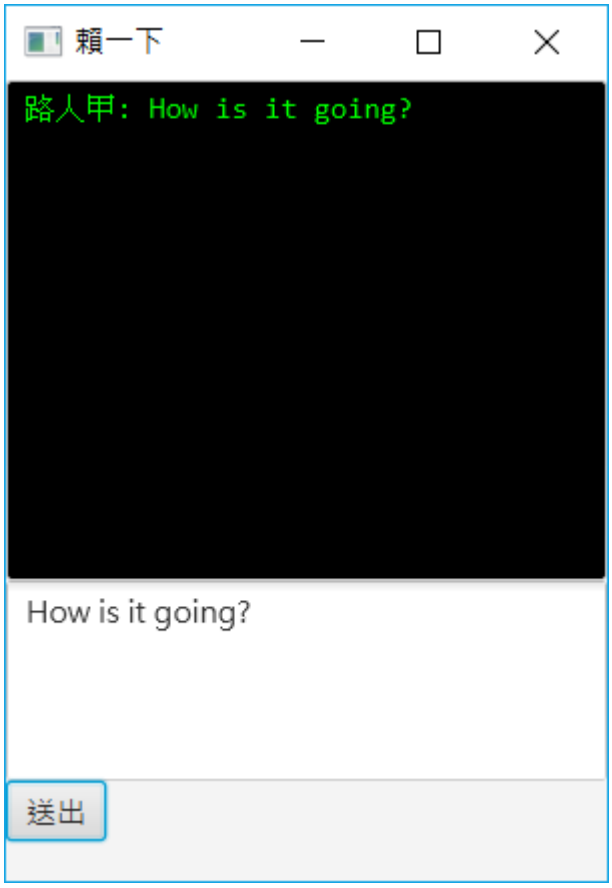
ClientFXMLController.java

```
package chapter31_chatting_tutorial;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;

public class ClientFXMLController implements Initializable {

    @FXML
    private TextArea display;
    @FXML
    private TextArea input;
    @FXML
    private Button btnSubmit;

    Socket socket = null;
    DataOutputStream toServer;
    DataInputStream fromServer;

    @FXML
    TextField textfield_name;
```

```
@FXML
private Button btnConnect;

@Override
public void initialize(URL url, ResourceBundle rb) {

}

@FXML
private void send(ActionEvent event) {

    try {
        //送出訊息給伺服器
        toServer.writeUTF(input.getText());
        toServer.flush();
    } catch (IOException e) {
        System.out.println("傳送訊息發生異常:" + e.toString());
        display.appendText("傳送訊息發生異常(斷線)\n");
    }
}

@FXML
private void connect2server(ActionEvent event) {

    if (textfield_name.getText().isEmpty()) {
        display.appendText("請輸入使用者代號...\n");
        return;
    }

    try {
        Socket socket = new Socket("localhost", 1024);
        fromServer = new DataInputStream(socket.getInputStream());
        toServer = new DataOutputStream(socket.getOutputStream());
        display.setText(textfield_name.getText() + "    連線成功...\n");
        //連線之後將連線按鈕 disable
        btnConnect.setDisable(true);

    } catch (IOException e) {
```

```
                display.setText("無法連線...\n");
                System.out.println("無法連線:" + e.toString());
            }

            //匿名方式產新一個新執行緒物件
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        //送出使用者名稱給伺服器
                        toServer.writeUTF(textfield_name.getText());

                        //連續監聽伺服器串流通道訊息
                        while (true) {
                            //讀入伺服器送過來的資訊
                            String msg = fromServer.readUTF();
                            display.appendText(msg + "\n");
                        }
                    } catch (IOException e) {
                        System.out.println("" + e.toString());
                    }
                }
            }).start();

        } //connect2server
}//class end
```

簡易版 Client 端(手工版，沒有用 FXML 設計)




```
package chapter31_chatting_tutorial;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class V31_Client extends Application {
```

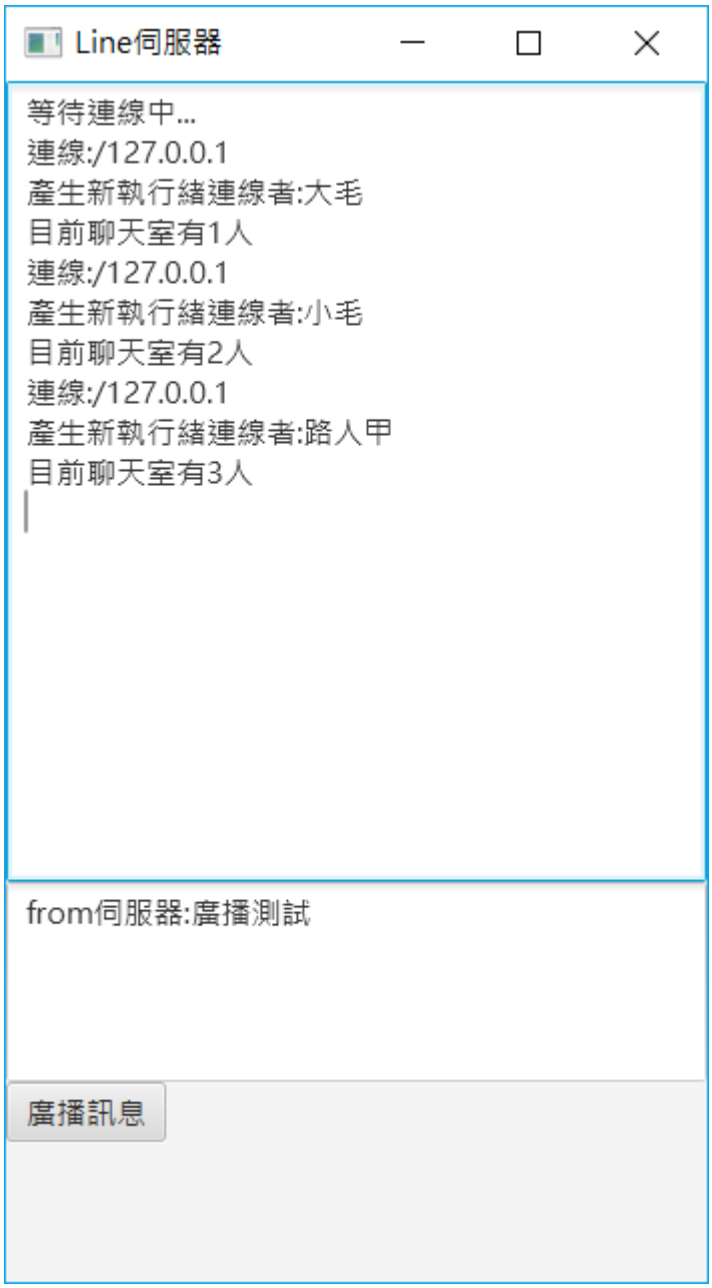
```
Socket socket = null;
DataOutputStream toServer;
DataInputStream fromServer;

Button btnSubmit = new Button("送出");
TextArea display = new TextArea();
String user_name = "路人甲";
TextArea input = new TextArea("How is it going?");

//建構子
//建構子會優先執行，之後再執行 public void start(Stage primaryStage){}
public V31_Client()
{
    //產生一個 Socket 物件-連線到伺服器
    try {
        Socket socket = new Socket("localhost", 1024);
        fromServer = new DataInputStream(socket.getInputStream());
        toServer = new DataOutputStream(socket.getOutputStream());
    } catch (IOException e) {
        System.out.println("無法連線:" + e.toString());
    }

    //匿名方式產新一個新執行緒物件-送出使用者名稱並連續監聽伺服器串流通道訊息
    new Thread(new Runnable() {
        @Override
        public void run() {
            try {

                //送出使用者名稱給伺服器
                toServer.writeUTF(user_name);

                //連續監聽伺服器串流通道訊息
                while (true) {
                    //讀入伺服器送過來的資訊
                    String msg = fromServer.readUTF();
                    display.appendText(msg + "\n");
                }
            } catch (IOException e) {
```

```
                    System.out.println("" + e.toString());
                }
            }
        }).start();
    }

    @Override
    public void start(Stage primaryStage) {
        FlowPane root = new FlowPane();
        display.setPrefSize(300, 250);
        input.setPrefSize(300, 100);
        display.setStyle(""
                + "-fx-control-inner-background:#000000; "
                + "-fx-font-family: Consolas; "
                + "-fx-highlight-fill: #00ff00; "
                + "-fx-highlight-text-fill: #000000; "
                + "-fx-text-fill: #00ff00; ");

        root.getChildren().add(display);
        root.getChildren().add(input);
        root.getChildren().add(btnSubmit);

        Scene scene = new Scene(root, 300, 400);

        primaryStage.setTitle("賴一下");
        primaryStage.setScene(scene);
        primaryStage.show();
        primaryStage.setOnCloseRequest(e -> {
            Platform.exit();
            System.exit(0);
        });


        btnSubmit.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                try {
                    //送出訊息給伺服器
```

```
                        toServer.writeUTF(input.getText());
                        toServer.flush();
                    } catch (IOException e) {
                        System.out.println("傳送訊息發生異常:" + e.toString());
                        display.appendText("傳送訊息發生異常(斷線)\n");
                    }
                }
            });
    }

    public static void main(String[] args) {
        launch(args);
    }

}
```

Server 端

Server.java




```
package chapter31_chatting_tutorial;


import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;
import javafx.application.Application;
```

```
import javafx.application.Platform;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class V31_Server extends Application {

    // 全域變數 整個程式有多個地方會用這些變數
    private final List<DataOutputStream> output2clients = new ArrayList();
    private TextArea display = new TextArea();

    public V31_Server() {

        //建立一個伺服器 ServerSocket 執行緒
        //也可以移至 start()中
        new ThreadServer().start();
    }

    @Override
    public void start(Stage stage) {

        Button btnSubmit = new Button("廣播訊息");
        TextArea input = new TextArea("from 伺服器:廣播測試");

        FlowPane root = new FlowPane();
        display.setPrefSize(350, 400);
        input.setPrefSize(350, 100);
        root.getChildren().add(display);
        root.getChildren().add(input);
        root.getChildren().add(btnSubmit);

        Scene scene = new Scene(root, 350, 600);

        stage.setTitle("Line 伺服器");
        stage.setScene(scene);
        stage.show();
```

```
        stage.setOnCloseRequest(e -> {
            System.exit(0); //結束程式
        });



        //建立一個伺服器 ServerSocket 執行緒
        //可以移至建構子
        // new ThreadServer().start();

    }

    public static void main(String[] args) {
        launch(args);
    }

    class ThreadServer extends Thread {

        private final int port = 1024;

        @Override
        public void run() {
            try {
                //建立一個伺服器 ServerSocket
                ServerSocket server = new ServerSocket(port);
                display.appendText("等待連線中...\n");

                while (true) {
                    //(1)等待有 client 連線
                    Socket socket = server.accept();

                    display.appendText("連線:" + socket.getInetAddress()+"\n");

                    //(2)建立連線之後，初始化一個 client 執行緒物件，
                    // 在執行緒裡面做詳細的訊息處理
                    // 若連線有 3 人，就會有 3 個 client 執行緒物件被產生
                    new ThreadClient(socket).start();
                }
```

```
            } catch (IOException e) {
                System.out.println("伺服器啟動異常:" + e.toString());
            }

        }
    }

    class ThreadClient extends Thread {

        private Socket socket; //連線到伺服器 Socket

        private DataInputStream fromClient;//進來的串流
        private DataOutputStream toClient;//出去的串流
        private String name; //連線者

        public ThreadClient(Socket socket) {
            this.socket = socket;
        }

        @Override
        public void run() {

            try {
                // 建立輸出與輸入串流
                fromClient = new DataInputStream(socket.getInputStream());
                toClient = new DataOutputStream(socket.getOutputStream());

                //讀取姓名
                name = fromClient.readUTF();
                display.appendText("產生新執行緒連線者:" + name + "\n");

                //把串流通道存放在連線集合中
                output2clients.add(toClient);
                display.appendText("目前聊天室有" + output2clients.size()+"人\n");

                //無窮迴圈 接收進來的訊息 並廣播出去
                //直到使用者結束連線，會跳去執行 finally 區塊
                while (true) {
```

```
                    //讀取使用者送來的訊息
                    String msg = fromClient.readUTF();

                    //使用者訊息很多，輸出到 terminal，或存到資料庫
                    //display.appendText(String.format("%s: %s\n", name, msg));
                    System.out.printf("%s: %s\n", name, msg);

                    //送出訊息給所有的通道
                    for (DataOutputStream writer : output2clients) {
                        writer.writeUTF(String.format("%s: %s", name, msg));
                        writer.flush();
                    }
                }

            } catch (IOException ex) {
                System.out.println("客戶端結束退出:" + ex.toString());
            } finally //最後關閉這個連線
            {

                try {
                    socket.close();
                    display.appendText("關閉連線:" + name + "\n");
                    output2clients.remove(toClient);
                    display.appendText("目前聊天室有" + output2clients.size()+"人\n");
                } catch (IOException ex) {
                    System.out.println("關閉連線異常:" + ex.toString());
                }
            }
        }
    }
}
```