|  |
| --- |
| PACIFIC LUTHERAN UNIVERSITY |
| Laboratory 2 |
| Echo Server Application |

|  |
| --- |
| Sheyla Trudo  3-9-2015 |

Table of Contents

[1. List of Figures i](#_Toc413834760)

[2. Introduction 1](#_Toc413834761)

[3. Requirements 1](#_Toc413834762)

[4. Design 1](#_Toc413834763)

[**Figure 1** 2](#_Toc413834764)

[**Figure 2** 3](#_Toc413834765)

[**Figure 3** 4](#_Toc413834766)

[5. Implementation 4](#_Toc413834767)

[List of Files 4](#_Toc413834768)

[Lab2Server.java 4](#_Toc413834769)

[Connection.java 5](#_Toc413834770)

[ServerController.java 5](#_Toc413834771)

[ServerView.java 5](#_Toc413834772)

[ServerModel.java 5](#_Toc413834773)

[6. Bibliography 6](#_Toc413834774)

# 1. List of Figures

Figure 1 - Design of the Server application user interface 2

Figure 2 - Multiple threading psuedo code provided by the lab overview documents. 3

Figure 3 - UML Diagram of the application. 4

# 2. Introduction

The purpose of this lab is to become familiar with Java ServerSocket communications. In this section of the lab, the idea is to write the server side implementation of a messenger application. The application is implemented in a simple GUI that allows the server administrator to input and send messages to all clients connected to the server. The server also echoes back messages from any client connected to the server to all the clients of the server.

# 3. Requirements

Below is a list of requirements that must be fulfilled by the application.

* Accept connections from clients
* Send user messages to clients
* Receive user messages from clients
* Provide a way for users to exchange messages.
* Each user’s message is delivered to all other users connected to the server.
* GUI application

Super-requirement additions.

* The application informs the server when a client has connected.
* A ‘Clear’ button allows the server administrator to clear all the text within the JTextArea, txtLog, that shows all communication between clients on the server.

# 4. Design

The design of the application includes a simple user interface with a message log that displays messages received from clients and shows when clients connect/disconnect from the server. The UI also includes an area for the user to enter and send messages to all users connected to the server.

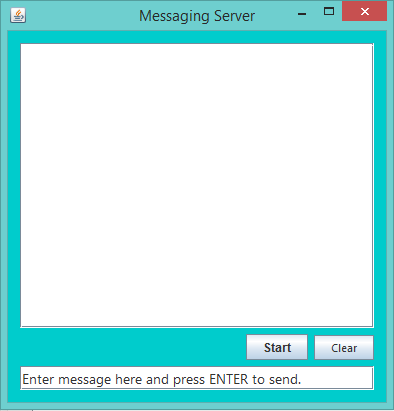


Figure 1 - Design of the Server application user interface

The pseudo code for the server side design was provided by Dr. George Hauser as part of the lab overview.

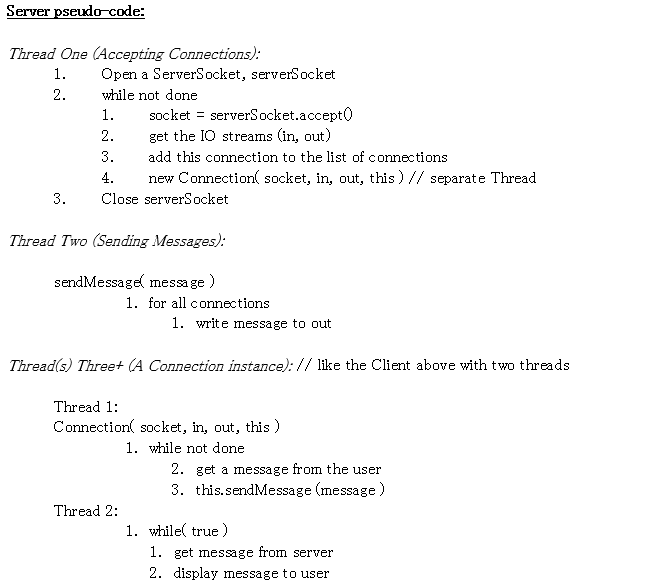


Figure 2 - Multiple threading pseudo code provided by the lab overview documents.

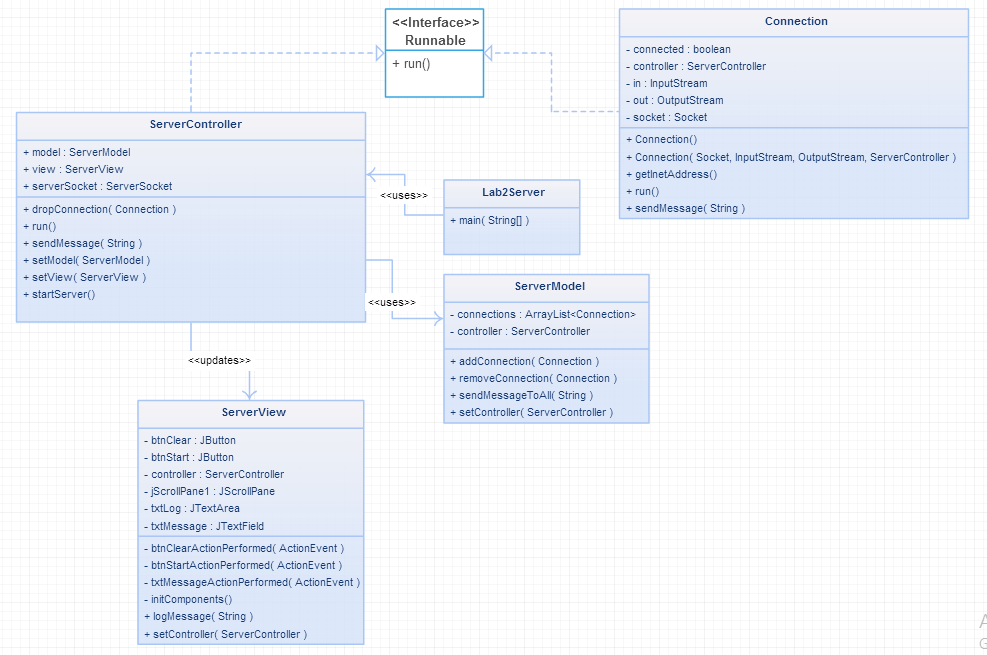


Figure 3 - UML Diagram of the application.

# 5. Implementation

List of Files within the project:

* Lab2Server.java
* Connection.java
* ServerController.java
* ServerView.java
* ServerModel.java

## Lab2Server.java

This particular class is in charge of instantiation and initialization of all objects within the program. It contains the main( String[] args ) class.

Creates:

* JFrame frame
* Serverview view
* ServerController controller
* ServerModel model

Sets

* frame DefaultCloseOperation
* view’s Controller to controller
* controller’s View to view
* controller’s Model to model
* model’s Controller to view

## Connection.java

This class is in charge of handling individual connections to the server. It opens a socket with an InputStream and an OutputStream. The Connection handles two separate threads. One thread listens for incoming messages and the other sends a message to the client through its OutputStream.

When a client disconnects, the reading thread ( run() ) throws an exception. This exception is handled by removing the Connection from the connection list in the ServerModel. The view is updated with a message saying “<Socket IP Address> has been disconnected”.

## ServerController.java

This class is in charge of multiple functions. One of them is to start the server broadcast on a thread. This allows clients to connect to the server by listening for new connections.

On another thread the controller handles messages received from separate Connection objects. The controller then passes on the message to the model to send out the message to all connected clients, as well as telling the view to update with the received message.

Another function of the controller is to handle the Connection disconnects. It does this by sending the dropped Connection object to the removeConnection() function in the model.

## ServerView.java

A JPanel that includes methods for handling user input. When the user presses enter after entering a message, the view sends the message to the controller to be handled. Also includes methods for updated the text in the message log.

## ServerModel.java

The main purpose of this class is to store the list of connections to the server. It contains methods for adding and removing connections, as well as for sending a String to each Connection currently connected to the server.

# 6. Bibliography

Dr. George Hauser