Practical 4 – TWIt2

Introduction

This practical required a Java implementation of a peer to peer instant messaging program. Peer to peer implies that each instance of the application will have to act as both a server and a client. Ideally, it also eliminates the need for a centralized server, but in most cases it is easier to have one, because otherwise the user would need to know the hostname or IP address of whomever they try to send messages to. Instant messaging usually refers to "sending real time messages to another Internet user" (Holetzky, 2010), but in the case of Windows Live Messenger (formerly named MSN Messenger), users can also send messages to friends who are offline. The approach taken in this practical, however, only allows messages to be sent between two online users. It is similar to IRC in that clients connect to a name-server and retrieve a list of online users, who they can then choose to connect to and send and receive messages. However, it does not allow chat between more than two users at the time in an IRC-like chat-room.

Class diagram

Due to Together's lack of cooperation, UML class diagrams have not been included. However, JavaDocs are supplied in the folder labelled JavaDocs.

Here is a brief explanation of the use of each class:

Class	Use
twit2.GUI	The main graphical user interface for the peer. Enables users to change settings, connect to a name server, set up their own local chat server and open chat windows with other peers.
twit2.KeepAlive	A Thread used to keep a connection alive (and avoid it timing out).
twit2.NameServerProfile	A NameServerProfile stores information about a name-server: a profile name, hostname and port number.
twit2.Peer	A wrapper used to store information about Peers to which a user can connect.
twit2.Server	The server-side of the peer. Accepts incoming chat connections.
twit2.ServerProfileDialog	Dialog used to edit and create server profiles.
twit2.UserListUpdater	Receives messages from the name server about users entering and leaving the server.
twit2.Validator	Validates input in ServerProfileDialog.
twit2.awt.HorizontalPanel	A horizontally-aligned panel.
twit2.awt.VerticalPanel	A vertically-aligned panel.
twit2.awt.Popup	Popups used to report information and errors.
twit2.chat.ChatWindow	The chat window used to chat to other peers.
twit2.chat.MessageReceiver	Deals with incoming messages from other peers. These are usually appended to the text-area in the ChatWindow.

twit2.io.ByteReader	Reads Strings from InputStreams.
twit2.io.ByteWriter	Converts strings to bytes which are sent written to OutputStreams.
twit2.nameserver.ClientHandler	The side of the name server which deals with connected peers.
twit2.nameserver.NameServer	The name server, as described later.
twit2.nameserver.NameServerException	If a problem is detected when initializing the connection with a peer, this exception is thrown.
twit2.nameserver.Runner	Runner class for the name server. Accepts quit-command.

The Name-server (TWIt2.nameserver.NameServer)

The name-server keeps track of who is online and sends the address and port number of users with a specified username upon request. More specifically, once a peer connects to the name server, it receives a list of all peers who are connected, with their nickname, address and the port number of their server. This name-server implementation is completely text-based. This decision was made because the end user of the name server is expected to be more experienced than the user of the chat client.

Sample output:

```
Setting up server...
Name server set up at port 60514
Enter QUIT to shut down the server.
Accepted connection from subhons-38-w.cs.st-andrews.ac.uk
Sending to all: ONL Ole;138.251.205.58;60516
Accepted connection from subhons-38-w.cs.st-andrews.ac.uk
Sending to all: ONL Pelle;138.251.205.58;60515
Pelle;subhons-38-w.cs.st-andrews.ac.uk;60515&Ole;subhons-38-w.cs.st-andrews.ac.uk;60516&
Accepted connection from subhons-38-w.cs.st-andrews.ac.uk
Sending to all: ONL Elli;138.251.205.58;60519
andrews.ac.uk;60516&Elli;subhons-38-w.cs.st-andrews.ac.uk;60519&
Accepted connection from subhons-38-w.cs.st-andrews.ac.uk
Sending to all: ONL Sophie;138.251.205.58;60520
Received quit command from Pelle
Sending to all: OFL Pelle
```

Meaning of server messages

Message	Meaning
ONL *nickname*;*hostname*;*portNumber*	A user with a given nickname, hostname and port number has connected to the name server.
OFL *nickname*	The user with the given nickname has left the nameserver.

These messages are dealt with by adding and removing users from a list in the peer to peer client's graphical user interface.

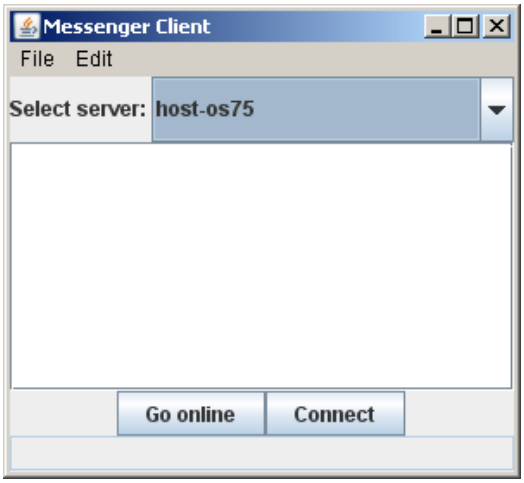
The Name-Server keeps connections open until it is shut down or the peers wish to leave the server.

The Peer (TWIt2.GUI)

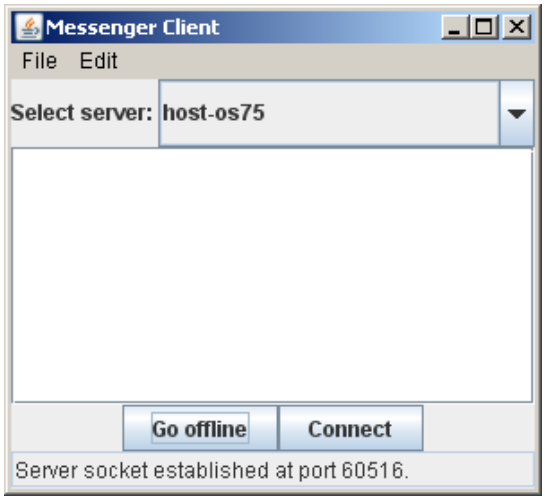
The peer is the application user by the general user. It has been given a graphical user interface in order to be more user-friendly. It connects to the name server, receives a list of people who are online and enables a user to open conversations with other users.

The Main Window

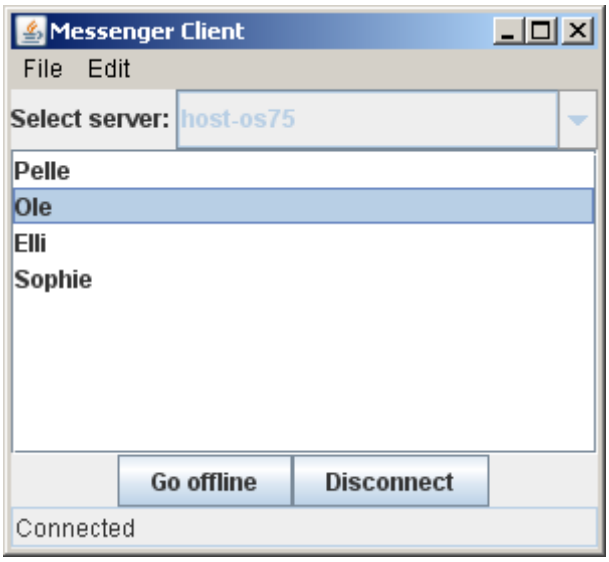
Initially, the main window looks like the one to the left below. In order to set up the chat server required to receive incoming conversations, the user has to click the "Go online" button.



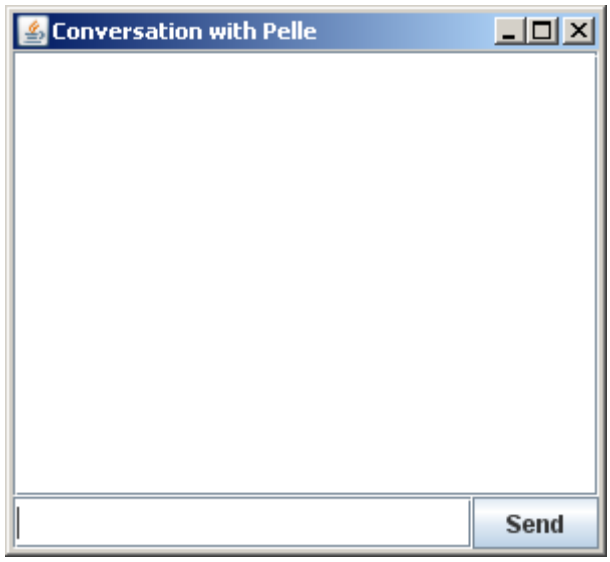
After the "Go online" button is pressed, the window looks like the one below, if the connection was successful. Now the connect button is pressed after a name server is selected using the checkbox at the top of the window.



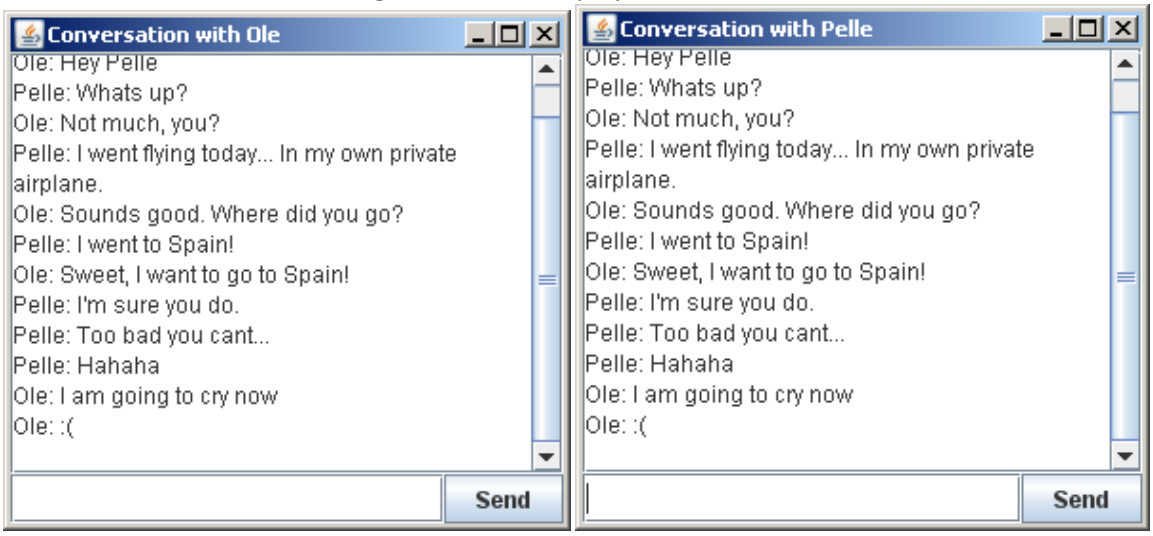
The user now receives a list of the users who are currently connected to the server. See picture below.



User can now double click one of the names on the list to start a conversation with that person:

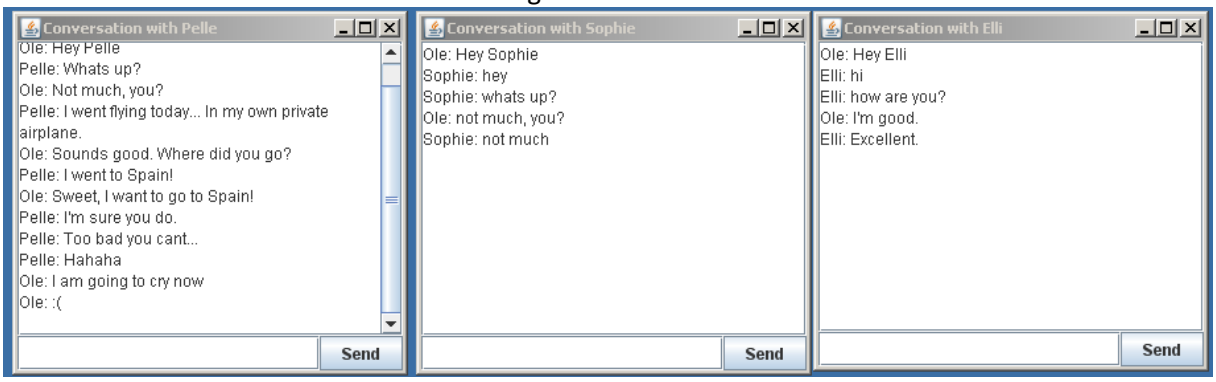


The users can now enter messages which are displayed in a text area:



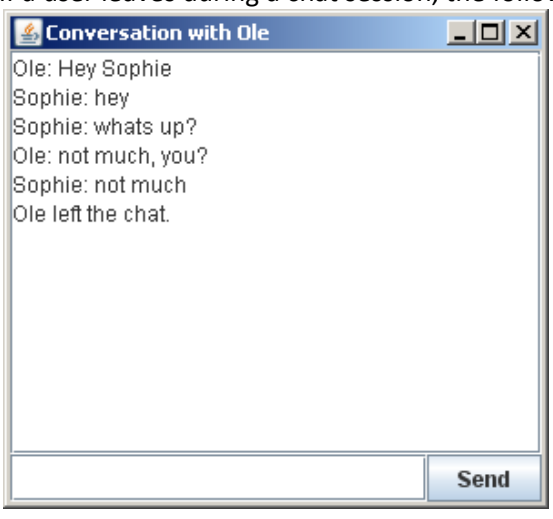
The messages are displayed to both users and if there is enough of them, a scroll bar appears on the side, allowing users to scroll up to see the older messages.

Users can have several conversations running at the same time.



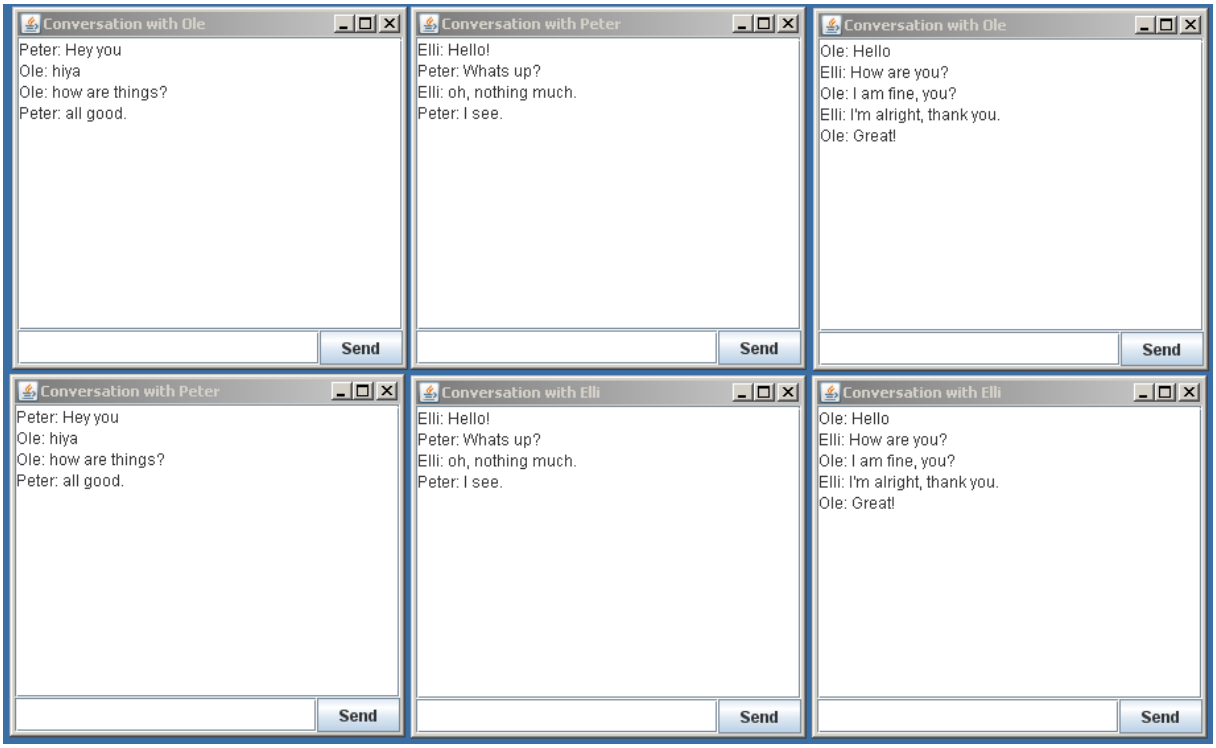
While these conversations are going on, these people may be chatting with other people as well.

If a user leaves during a chat session, the following message is sent to the interacting peer:



Most of the testing was done only using one machine, but it does also work over the network. In the cases where testing was done over the network, the server was run over SSH.

Chat Window (TWIT2.chat.ChatWindow)



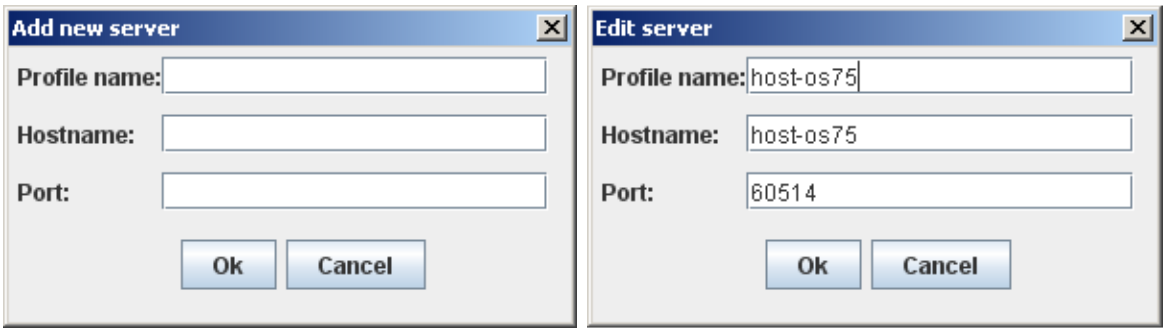
The server allows multiple chat sessions to be run in simultaneously with different people. The first two are both representing a conversation between “Ole” and “Peter”, the second vertical pair is a conversation between “Elli” and “Peter” and the third is between “Ole” and “Elli”.

Settings

Name-server profiles

Name server profiles can be selected using the checkbox in the main window. They can also be edited and created through use of the Edit-menu.

The following windows then appear when the Edit and new server menu-items are pressed:



The instructions on the screen are easy to understand, and therefore user-friendly. If the implementation were to be text-based, it may not have been as clear to the user how to add a new name-server.

Borrowed code

MouseAdapter is used in order to make a ChatWindow appear when a user double-clicks the list. This idea was taken from sample code from *Double click on a JList* (Gagnon).

Conclusion

To be fair, this practical has not taught me as much as previous ones. I was already familiar with most of the concepts required. However, it has taught me that you have to be extremely careful with what you do, because there are a lot of things that can go wrong. For instance, after I made some changes the chat windows stopped appearing when I double-clicked nicknames from the list. I thought there was something wrong with the connection, but it ended up being a problem with a missing setVisible(true) in the ChatWindow class.

In the future a more sophisticated login system could be implemented, where users register and have set passwords. This would make sure two different people could not have the same username at different times. Additionally, there are exceptions which could be dealt with in better ways, and connections which are kept open throughout, rather than reopened upon demand. This is, according to demonstrators in the lab, bad practice and should be avoided.

Bibliography

Gagnon, R. (n.d.). *Double click on a JList*. Retrieved April 14, 2010, from Real's Java How-to: http://www.rgagnon.com/javadetails/java-0219.html

Holetzky, S. (2010, March 16). *What is Instant Messaging?* Retrieved April 13, 2010, from wiseGEEK: http://www.wisegeek.com/what-is-instant-messaging.htm