Object Oriented Design and Programming (CS206)

Mini-Project Report

Arun Jose
S4 CSE
Roll No. 12

Computer Science and Engineering
College of Engineering Trivandrum
April 2020

Contents

1 Aim

To share information between two entities - Client, and Server. A client can chat or send messages to other clients. Messages can be sent to one or multiple other clients.

2 Introduction

The project consists of the client Applet, and the Server. The server handles messages being passed to and from various clients through a port, and the applet facilitates a front-end GUI as an alternative to the terminal for allowing clients to pass messages.

3 Code

3.1 ChatApplet.java

```
/*
<applet code="ChatApplet" port=3000 width=300 height=400>
</applet>
*/

import java.awt.*;
import java.applet.Applet;
import java.net.*;
import java.io.*;
import java.awt.event.*;
public class ChatApplet extends Applet implements Runnable
{
 /** Initialize the applet and set up the GUI.
 */
 public void init()
 { setLayout(new BorderLayout());
 messages = new TextArea();
 messages.setFont(new Font("Courier", Font.PLAIN, 10));
 add("Center", messages);

 send = new Button("Send a Line of Text");
 Sender sender = new Sender();
 send.addActionListener(sender);
 add("South", send);

 entry = new TextField();
 entry.addActionListener(sender);
 add("North", entry);
```

```
messages.setText("Chat contents displayed here.");
try
{
    URL server = getDocumentBase();
    int port = Integer.parseInt(getParameter("port"));
    connection = new Socket(server.getHost(),port);
    inputStream = new BufferedReader(
                  new InputStreamReader(connection.getInputStream()));
    outputWriter = new PrintWriter( connection.getOutputStream(),true);
    }
catch(UnknownHostException e){messages.append("No Host.");}
    catch(IOException e){messages.append("Startup I/O Error: " +e);}

    new Thread(this).start();
   entry.setText("Type your messages here.");
   entry.selectAll();
  entry.requestFocus();
setVisible(true);
}

/** Return the size of the frame in which the applet is displayed
* @return The size of the Frame.
*/
public Dimension setSize() {return new Dimension(600, 400);}

public void run()
{ try
{ while (true)
{ String fromServer = inputStream.readLine();
if (fromServer != null)
{ fromServer += '\n' ;
messages.append(fromServer);
entry.requestFocus();
}
else break;
}
    }
    catch(IOException e){messages.append("Fetch Error: " +e);}
}

/** Retrieve information about the applet
* @return The information String.
```

```
 */
 public String getAppletInfo()
 { return "JBChat v0.2, Joseph Bergin";
 }

 /** Break the communication link.
 */
 public void destroy()
 { try
 { outputWriter.println("BYE");
 connection.close();
 }
 catch(IOException e){}
 }

 private TextField entry;
 private TextArea messages ;
 private Button send;
 private BufferedReader inputStream;
 private PrintWriter outputWriter;
 private Socket connection;

 private class Sender implements ActionListener
 { public void actionPerformed(ActionEvent e)
 { // Only two sources with identical actions
 String s = entry.getText();
 if(s != null && s.trim().length()>0)
 { outputWriter.println(s);
 entry.setText("");
 }
 entry.requestFocus();
 }
 }

}
```

3.2 ChatServer.java

```
import java.io.*;
import java.net.*;

public class ChatServer
{
```

```
public static void main(String[] args )
{ int i;
int socketNumber = 4440;
if (args.length > 0)
{ size = Integer.parseInt(args[0]);
}
if (args.length > 1)
{ socketNumber = Integer.parseInt(args[1]);
}
try
{ ServerSocket s = new ServerSocket(socketNumber);
for(i= 0; i < size; ++i)
{ sessions[i] = null;
}
new Echoer().start();
while(true)
{ Socket incoming = s.accept( );
boolean found = false;
int numusers = 0;
int usernum = -1;
synchronized(sessions)
{ for(i = 0; i < size; ++i)
{ if(sessions[i] == null)
{ if(!found)
{ sessions[i] = new PrintStream(incoming.getOutputStream());
new ChatHandler(incoming, i).start();
found = true;
usernum = i;
//System.out.println("assign "+i);
}
}
else numusers++;
}
if(!found)
{ PrintStream temp = new PrintStream(incoming.getOutputStream());
temp.println("\n No available entry.  Disconnecting. Sorry. \n");
temp.println("\n You must reload to try again. \n");
temp = null; // Permit garbage collection of the PrintStream.
s.close();
}
else
{ sessions[usernum].println("\nThere are "+numusers+" other users.\n");
}
```

```
}
}
}
catch (Exception e)
{ System.err.println("Error in main: " + e);
}
}

private static int size = 4;
private static PrintStream [] sessions = new PrintStream[size];
private static Buffer message = new Buffer();


private static class ChatHandler extends Thread
{ private Socket incoming;
private int counter;
ChatHandler(Socket i, int c) { incoming = i; counter = c; }

public void run()
{ try
{ BufferedReader in = new BufferedReader(
                        new InputStreamReader(incoming.getInputStream()));
String name = "";
synchronized(ChatServer.sessions)
{ PrintStream out = ChatServer.sessions[counter];
out.println("Enter your name: ");
name = in.readLine();
out.println( "Hello there, "+name+". Enter your name to exit.\n" );
}
boolean done = false;
while (!done)
{ String str = in.readLine();
if (str == null)
{ done = true;
}
else
{ ChatServer.message.set("(" + name + "): " + str + "\r");
if (str.trim().equals(name))
{
ChatServer.message.set(name + " is going on a journey far away now.");
done = true;
}
}
```

```
}
incoming.close();
}
catch (Exception e)
{ System.err.println("ChatHandler error: " + e);
}
synchronized(ChatServer.sessions)
{ ChatServer.sessions[counter].close();
ChatServer.sessions[counter] = null;
}
}
}// class ChatHandler

private static class Echoer extends Thread // One Reader of message.
{ // Broadcasts all messages to all users.
public void run()
{ while(true)
{ String s = ChatServer.message.get();
synchronized(ChatServer.sessions)
{ for(int i = 0; i < ChatServer.size; ++i)
if(ChatServer.sessions[i] != null)
ChatServer.sessions[i].println(s);
}
}
}
} // class Echoer

private static class Buffer  // Implements a shared one element queue.
{ private String message = null;
public synchronized void set(String s)
{ try
{ while(message  != null)
wait();
}
catch(InterruptedException e) {}
message = s;
notify();
}

public synchronized String get()
{ try
{ while( message == null)
wait();
```

```
 }
 catch(InterruptedException e) {return null;}
 String result = message;
 message = null;
 notify();
 return result;
 }
 } // class Buffer

}
```

4 Running the program

4.1 Chat

First, compile the server java code. In the main directory,

```
javac Server/ChatServer.java
```

Then run the server, passing as arguments both the limit on the number of participants at a time, and the localhost port.

```
java Server/ChatServer.java 4 3000
```

where 4 is the maximum number of participants, and 3000 is the port number it can be accessed through.

Then, in a new terminal, run the following code in the main directory:

```
telnet localhost 3000
```

This should begin an instance of the chat, and running simultaneous such terminals will allow you to pass messages through the server.

4.2 Applet

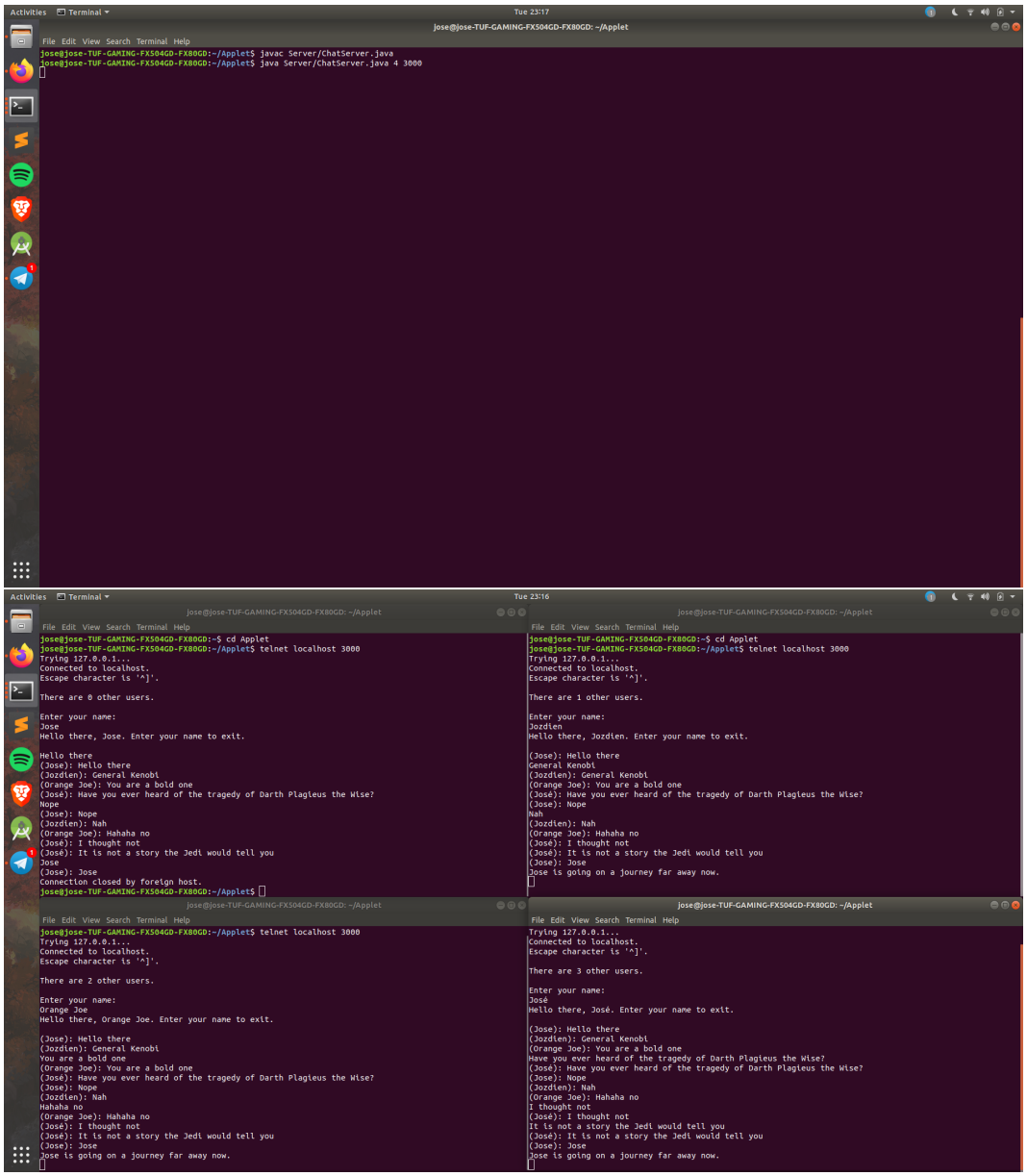
To run the messaging through an applet, first ensure that the functional dependancies have been met (primarily using Java 1.7 or lower). Then, compile the applet java code in the main directory as,

```
javac Client/ChatApplet.java
```

Then view the applet by running the following code:

```
appletviewer Client/ChatApplet.java
```

5 Messages Being Sent



6 Result

A messaging service was developed using Java Applets, and was tested on localhost to facilitate communication between multiple users at a port.