Distributed Systems

A Java based

Chat Application

Using JMS



Student Name : David Martin

Student Number : A00128473

MSc. Software Engineering

Table of Contents

[Introduction 3](#_Toc428123637)

[The JMS Message Object 4](#_Toc428123638)

[The Main Operation of the Chat Application 5](#_Toc428123639)

[Managing the “Users Online” TextArea by Use of a Timer 7](#_Toc428123640)

[How Users are handled when they join a chat session 7](#_Toc428123641)

[How Users are handled when they leave a chat session 9](#_Toc428123642)

[The removeUser() Method 11](#_Toc428123643)

[Additional Features 13](#_Toc428123644)

[Automatic Scrolling on the Message Received Area 13](#_Toc428123645)

[KeyListener on the message Textfield 14](#_Toc428123646)

[Clear Chat Option 15](#_Toc428123647)

[Backup Chat Option 15](#_Toc428123648)

[Project Source Code 16](#_Toc428123649)

[myChatGui.java 16](#_Toc428123650)

[LoginFrame.java 20](#_Toc428123651)

[Details.java 21](#_Toc428123652)

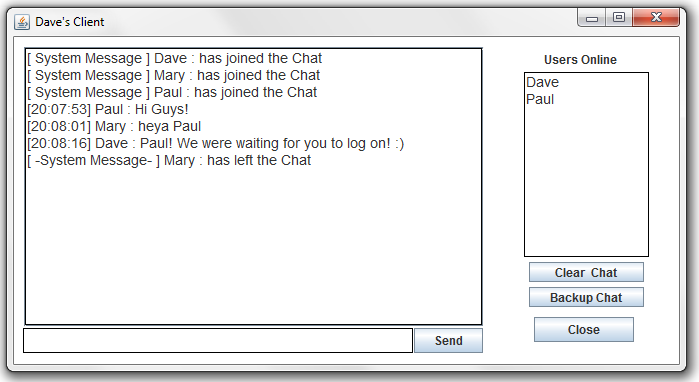
[References 22](#_Toc428123653)

# Introduction

This Document contains a brief description, outlining the main features of the Java Chat GUI application I built by the use of Java Messaging Service.

The main areas discussed are:

* **The main operation of the Chat Application**
  + The LoginFrame
  + Using a message object to record **User, Sent Time and the Message text**
  + Managing the “**Users Online” TextArea** by Use of a Timer
  + How Users are handled when they **join** a chat session
  + How Users are handled when they **leave** a chat session
  + Automatic **Scrolling** on the Message Received Area
  + **KeyListener** on the message Textfield
* **Additional Features**
  + **Clear** Chat Option
  + **Backup** Chat Option



# The JMS Message Object

This section discusses the JMS chat object that is used in the chat application. A class called “Details” was created which contains the elements shown in the class diagram below:

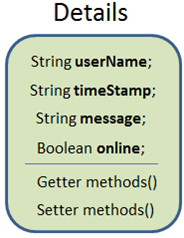


Figure 1. The JMS Object Class Diagram

**username**

* This String is used hold the chat alias that the User enters on the Login Frame.

**timeStamp**

* This String is used hold the timestamp at which the message object was generated

**message**

* This String is used hold the chat message entered by the User.

**online**

* This Boolean will identify if the User who’s client created the message object, is currently online. The purpose of this variable is to enable the application to identify when the D:\usersOnline.txt file needs to be updated to reflect a User joining or leaving the chat session.

The Code Snippet in Figure 2 shows the generation of a JMS Message object being instantiated on the event of the User clicking the “Send” button on the main chat frame.

On this event, a new message object is created, passing various information as arguments to the message object constructor as follows:

* the current Users chat alias
* the current timestamp
* the message entered by the user in the outgoingMessage textfield
* a Boolean value of true to reflect that this user is currently online.

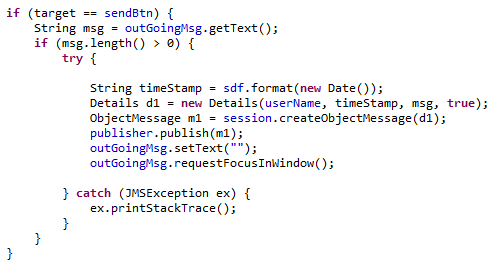
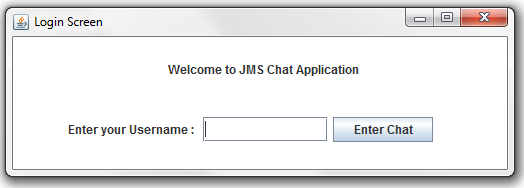
s

Figure 2. Code Snippet Of JMS a Message Object Being Created When the Send Button Is Pressed

# The Main Operation of the Chat Application

* The LoginFrame
* Using a message object to record User, Sent Time and the Message text
* How users are managed when they join or leave the chat
* Automatic Scrolling on the Message Received Area
* KeyListener on the message Textfield

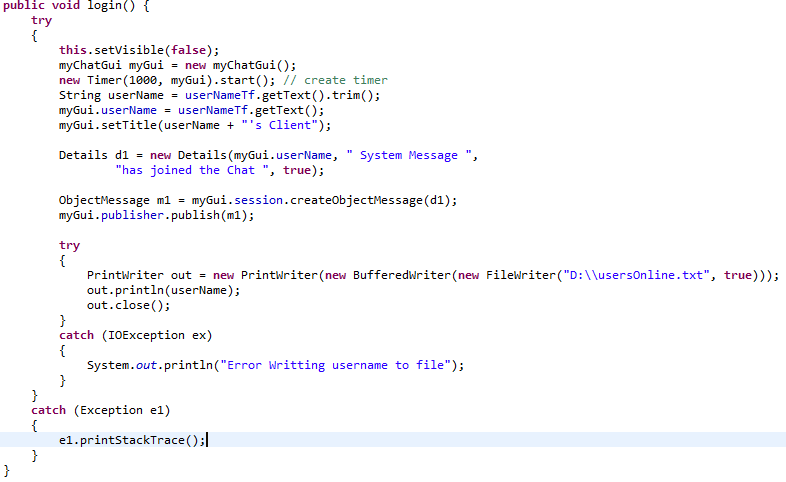
The Chat Application is initiated by the execution of the **LoginFrame.java** source file.



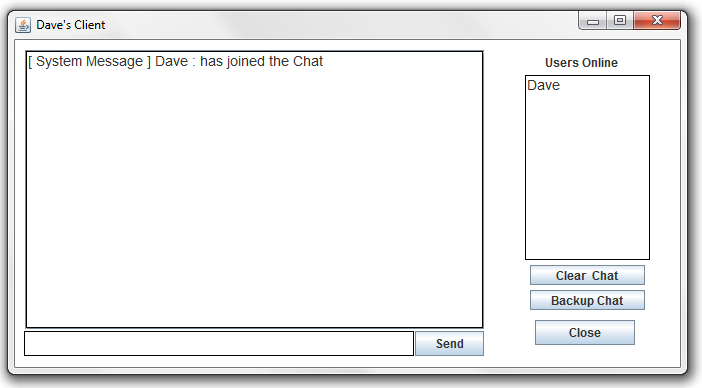
*Figure2. The Login Screen Frame*

As shown in figure 3, this frame prompts the user to enter the alias they would like to be displayed as their chat name. Once the alias is entered and the “Enter Chat” button is clicked, the login() method ( as shown in Figure 4 ) will be called by which the following sequence of events will executed:

* The Login Frame will be set to invisible.
* A new instance of the myChatGui class will be created called myGui
* A Timer will be added to the myGui frame.
* The Alias entered by the User will be passed to the myGui and set as the frames title.
* A JMS Chat message object will be created using the alias entered which will be published to the JMS server to broadcast to all current chat session clients, that a new user has entered the chat as a System Message as shown in Figure 5.
* Using PrintWriter, the Users chat alias will be appended to the file located at D:\usersOnline.txt.



*Figure 4. The Login() Method Code Snipped*

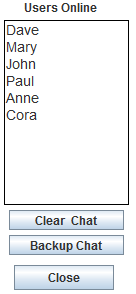


*Figure 5. The Main Chat Frame*

# Managing the “Users Online” TextArea by Use of a Timer

## How Users are handled when they join a chat session

When a new instance of the chat client is initiated from the Login frame, by the use of PrintWriter, the Users chat alias will be appended to a file located at D:\usersOnline.txt. If this file does not currently exist in that directory it will be created automatically.

.

*Figure 6. The Users Online Text Area*

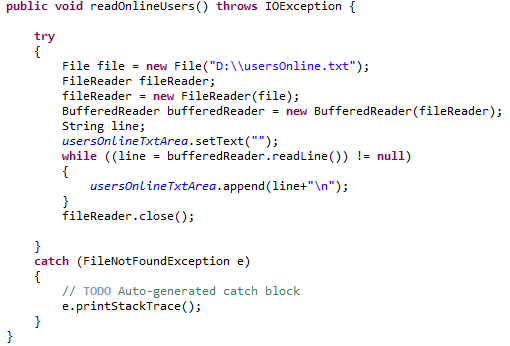
As mentioned in the previous section, on creation of the main chat frame, a timer with a duration of 1000 milliseconds is implemented to assist with the management of Username in the “Users Online” TextArea as shown in Figure 6.

For each tick of the Timer that resides on the main chat frame, the method readOnlineUsers() is called.

This method will perform the following sequence of events :

* A FileReader object will be created which points to the file D:\usersOnline.txt.
* A BufferedReader will then be create passing the FileReader object as an argument
* All information that currently is displayed in the usersOnlineTxtArea will be removed.
* By the utilisation of a While Loop, the usersOnlineTxtArea will be repopulated with the User alias’s that current exist in the D:\usersOnline.txt. file.

A Code Snippet of the readOnlineUsers() is shown in Figure 7 below.



*Figure 7. readOnlineUsers Method Code Snippet*

This proccess allows the usersOnlineTxtArea to be updated on each tick of the timer, displaying all usernames that are stored on the usersOnline.txt file on this TextArea on each interval.

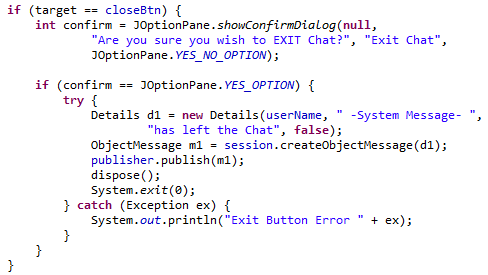
## How Users are handled when they leave a chat session

When a User decides to leave the current chat session by clicking the “Close” button, the follow sequence of events will take place (Figure 8):

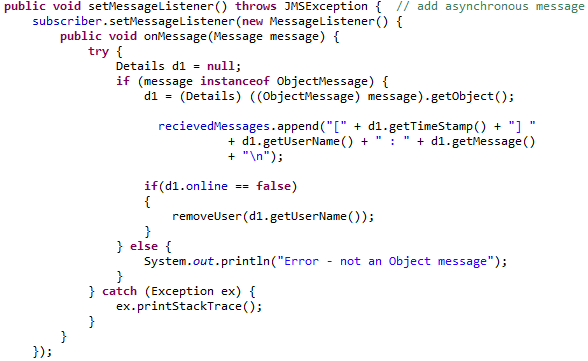
A confirm dialog box will appear asking the User if they are sure they wish to exit the chat session.

If the user confirms this action by click the “Yes” option, a new message object is created with parameters passed to generate a message.

**NB.** The Boolean field called “*online*” of this message object will be set to **false**, to reflect that this user is now offline.



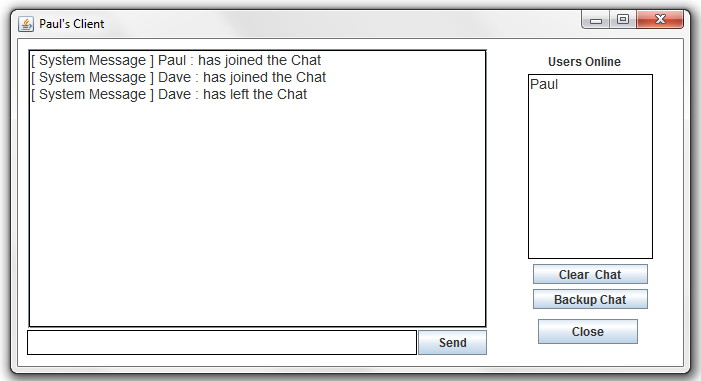
*Figure 8. Close Button Code Snippet*



*Figure 9. onMessage() method Code Snippet*

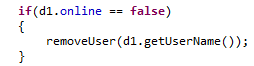
When this message object is handled by the onMessage method (Figure 9), the following events will happen:

* The message object generated by the Close Button event will be published to all current chat session clients, notifying them that the User has now left chat, see Figure 10.



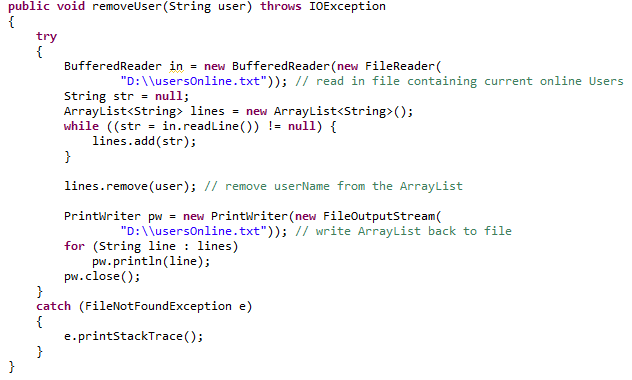
*Figure 10. Message Object published when a User has left the Chat Session*

* An If statement will then check to see if the boolean flag variable “online” is set to false, in this case, this variable is set to false, hence qualifying the conditions of the if statement. A method removeUser() is then called within the code block of this If statement passing the username field of the message Object as a parameter, see Figure 11.

.

*Figure 11. If Statement within the onMessage() method*

## The removeUser() Method

****Figure 12.removeUser() method Code Snippet

When the removeUser() method is called (Figure 12), by the use of BufferedReader and FIleReader, the text file residing at D:\usersOnline.txt is opened and its contents are then read in to an ArrayList called “lines”.

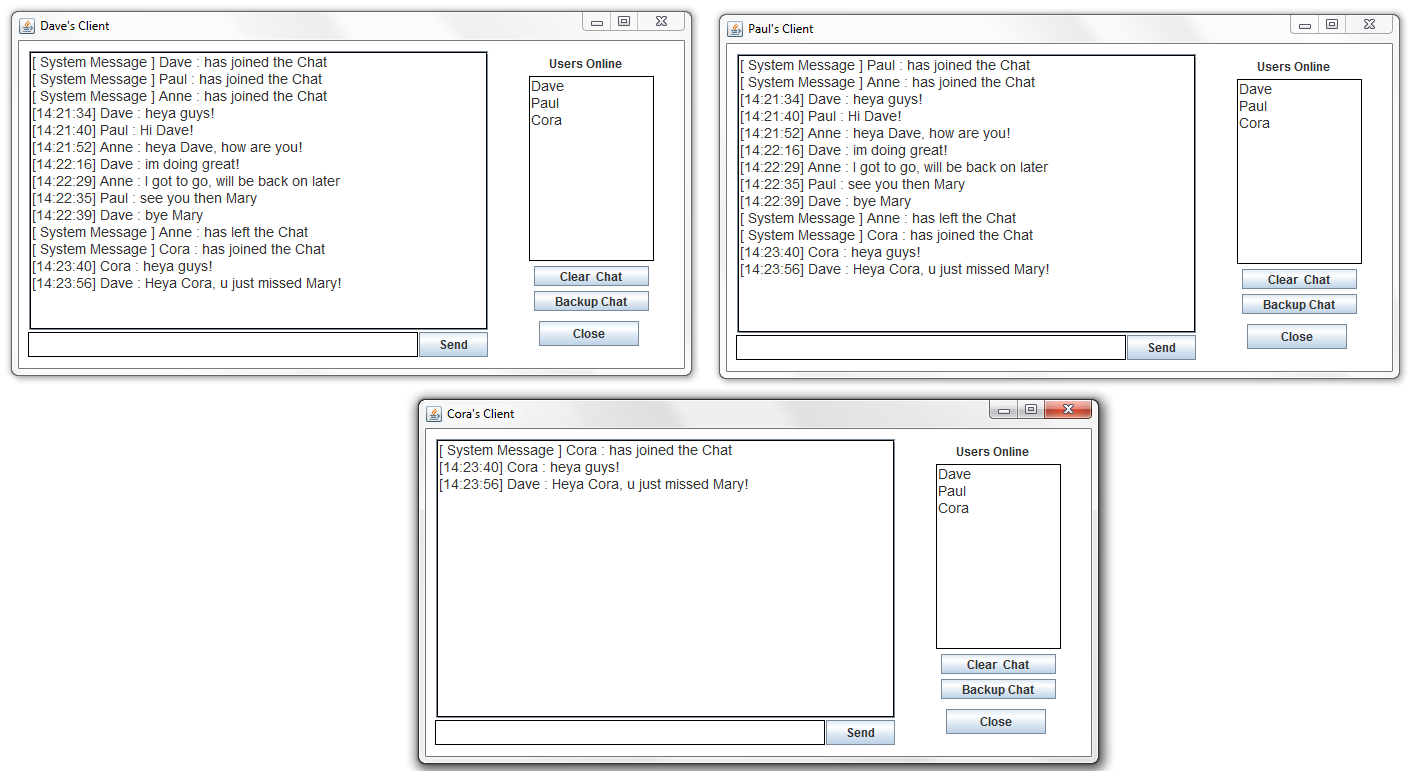
The .remove function this then used on this ArrayList passing the String containing the Username who has left the chat session as a parameter.

Once the Username has been removed from the ArrayList, by the use of PrintWriter, the List is then read back into the D:\usersOnline.txt file line by line.

D:\usersOnline.txt now contains all current chat Usernames, minus the Username of the client that just left the chat session.

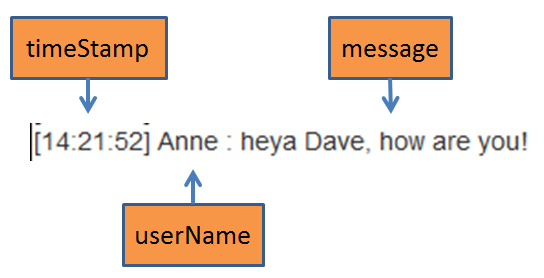
On the next tick of the timer that is on the myChatGui frame, the readOnlineUsers() method will be called, updating the onlineUsers TextArea to reflect that the User in question is no longer in the chat session.

Figure 13 shows an example of multiple instances of the chat client being used to chat by 3 users:



*Figure 13. Three Users Chatting*

The above demonstration shows 3 Users logged on to the Chat application. Every time a new instance of the chat client is opened by a different user, a message object is broadcasted as a system message to let all current users of the application know that a new user (identified by his alias) has entered the chat.



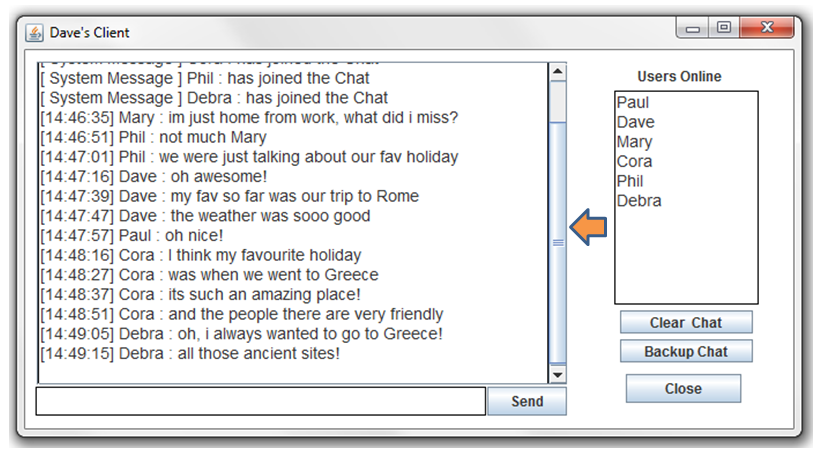
*Figure 14. A Message Object Displayed in the Chat text area*

We can also see that when a user sends a chat message, the time for which the message was sent is displayed along with the User’s alias name and then their chat message. In this instance, the user Anne, sent a message at 14:21:52, saying “heya Dave, how are you!”. As the code snippet in Figure 14 shows, this was achieved by including the user’s alias, the current system time and the chat text in a message object, see Figure 14.

# Additional Features

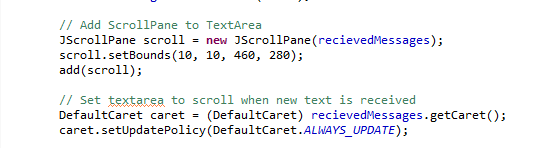
## Automatic Scrolling on the Message Received Area

As demonstrated in Figure 15, once a certain amount of messages have been received to the chat area, a scroll bar will appear which will automatically scroll to the bottom of the textarea once a new message has been received. The Code Snippet in Figure 9 shows how this was implemented.



*Figure 15. Automatic Scrolling on the Message Received Area*

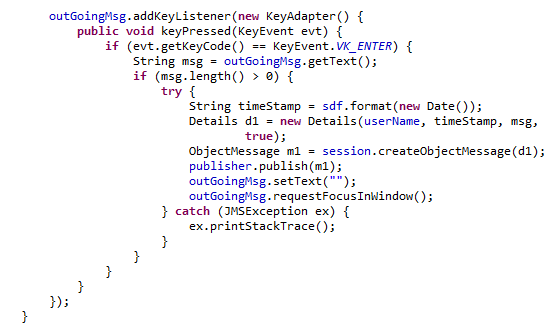
This automatic scrolling was implemented by adding a JScrollPane to the JTextArea that is used to receive chat messages. A caret was then implemented to automatically scroll to the bottom of the textarea once a new chat message is received. See Figure 16.



*Figure 16. Implementing the ScrollPane and Caret*

## KeyListener on the message Textfield

To add “*quality of life*” improvements for users of this application, I implemented a Keylistener which allows the User to simply press the enter key to send their message once they have typed it in, as opposed to having to constantly click the “Send” button for each message they want to broadcast. See Figure 17.



*Figure 16. Implementing KeyListener on the outGoingMessage TextField*

Once the user sends his/her message either by hitting the Enter key or clicking on the Send Button, the message textfield will be cleared and then set as focus again, allowing the User to conveniently continue to type a new message without being interrupted. See Figure 17.



*Figure 17. Setting the outGoingMessage to focus once a message is sent*

## Clear Chat Option

****

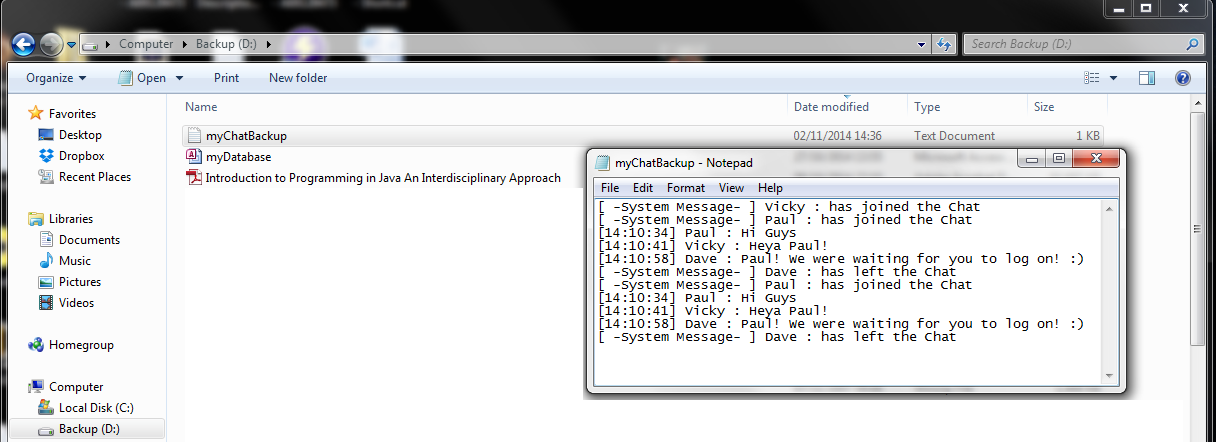
When the User clicks this button, all text from the JTextArea will be will be removed enabling the chat area to be cleared.

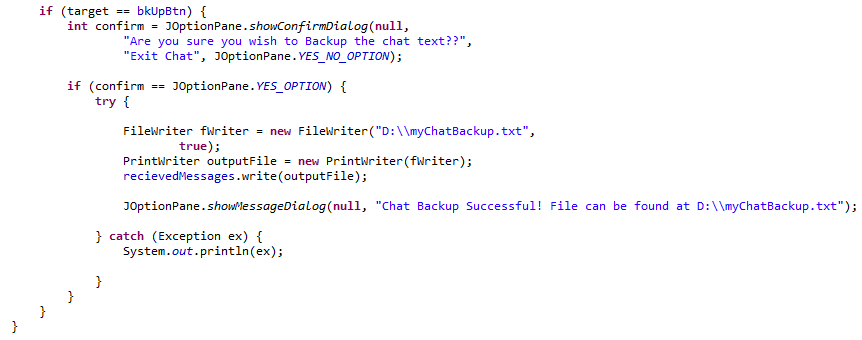


## Backup Chat Option

****

This feature allows the User to back up the current chat session to a text file that will be created and stored at the following path : **D:\myChatBackup.txt**, as demonstrated below.





# Project Source Code

## myChatGui.java

public class myChatGui **extends** JFrame **implements** ActionListener **{**

private static final long serialVersionUID **=** 1L**;**

TopicSubscriber subscriber**;**

TopicPublisher publisher**;**

TopicSession session**;**

String userName **=** ""**;**

String user**;**

private JTextArea recievedMessages**;**

private JTextField outGoingMsg**;**

private JButton sendBtn**;**

private JButton clearBtn**;**

private JButton bkUpBtn**;**

private JButton closeBtn**;**

private JLabel timeLbl**;**

private JLabel dateLbl**;**

private JLabel imageLbl**;**

public static JTextArea usersOnlineTxtArea**;**

private JLabel jcomp10**;**

Border border **=** BorderFactory**.**createLineBorder**(**Color**.**BLACK**);**

SimpleDateFormat sdf **=** **new** SimpleDateFormat**(**"HH:mm:ss"**);**

public myChatGui**()** **throws** Exception **{**

Hashtable**<**String**,** String**>** properties **=** **new** Hashtable**<**String**,** String**>();**

properties**.**put**(**Context**.**INITIAL\_CONTEXT\_FACTORY**,**

"org.exolab.jms.jndi.InitialContextFactory"**);**

properties**.**put**(**Context**.**PROVIDER\_URL**,** "rmi://localhost:1099/"**);**

Context ctx**;**

ctx **=** **new** InitialContext**(**properties**);**

TopicConnectionFactory factory **=** **(**TopicConnectionFactory**)** ctx

**.**lookup**(**"JmsTopicConnectionFactory"**);**

TopicConnection conn **=** factory**.**createTopicConnection**();**

session **=** conn**.**createTopicSession**(false,** Session**.**AUTO\_ACKNOWLEDGE**);**

final javax**.**jms**.**Topic topic **=** **(**javax**.**jms**.**Topic**)** ctx**.**lookup**(**"topic1"**);**

// create publisher and subscriber

publisher **=** session**.**createPublisher**(**topic**);**

subscriber **=** session**.**createSubscriber**(**topic**);**

// construct GUI components

recievedMessages **=** **new** JTextArea**();**

outGoingMsg **=** **new** JTextField**(**5**);**

sendBtn **=** **new** JButton**(**"Send"**);**

clearBtn **=** **new** JButton**(**"Clear Chat"**);**

bkUpBtn **=** **new** JButton**(**"Backup Chat"**);**

closeBtn **=** **new** JButton**(**"Close"**);**

timeLbl **=** **new** JLabel**(**""**);**

dateLbl **=** **new** JLabel**(**""**);**

usersOnlineTxtArea **=** **new** JTextArea**(**5**,** 5**);**

jcomp10 **=** **new** JLabel**(**"Users Online"**);**

ImageIcon image **=** **new** ImageIcon**(**"src\\java1.png"**);**

imageLbl **=** **new** JLabel**(**""**,** image**,** JLabel**.**CENTER**);**

recievedMessages**.**setFont**(new** Font**(**"Arial"**,** Font**.**PLAIN**,** 14**));**

usersOnlineTxtArea**.**setFont**(new** Font**(**"Arial"**,** Font**.**PLAIN**,** 14**));**

recievedMessages**.**setBorder**(**BorderFactory**.**createCompoundBorder**(**border**,**

BorderFactory**.**createEmptyBorder**(**1**,** 1**,** 1**,** 1**)));**

outGoingMsg**.**setBorder**(**BorderFactory**.**createCompoundBorder**(**border**,**

BorderFactory**.**createEmptyBorder**(**1**,** 1**,** 1**,** 1**)));**

usersOnlineTxtArea**.**setBorder**(**BorderFactory**.**createCompoundBorder**(**border**,**

BorderFactory**.**createEmptyBorder**(**1**,** 1**,** 1**,** 1**)));**

// adjust size and set layout

setSize**(**681**,** 365**);**

setLayout**(null);**

// add components

add**(**recievedMessages**);**

add**(**outGoingMsg**);**

add**(**sendBtn**);**

add**(**clearBtn**);**

add**(**bkUpBtn**);**

add**(**closeBtn**);**

add**(**timeLbl**);**

add**(**dateLbl**);**

// add(imageLbl);

add**(**usersOnlineTxtArea**);**

add**(**jcomp10**);**

setMessageListener**();**

conn**.**start**();**

// set component bounds (only needed by Absolute Positioning)

recievedMessages**.**setBounds**(**10**,** 10**,** 450**,** 280**);**

outGoingMsg**.**setBounds**(**9**,** 291**,** 390**,** 25**);**

sendBtn**.**setBounds**(**400**,** 291**,** 69**,** 25**);**

clearBtn**.**setBounds**(**515**,** 225**,** 115**,** 20**);**

bkUpBtn**.**setBounds**(**515**,** 250**,** 115**,** 20**);**

closeBtn**.**setBounds**(**520**,** 280**,** 100**,** 25**);**

imageLbl**.**setBounds**(**505**,** 45**,** 130**,** 155**);**

usersOnlineTxtArea**.**setBounds**(**510**,** 35**,** 125**,** 185**);**

jcomp10**.**setBounds**(**530**,** 10**,** 100**,** 25**);**

usersOnlineTxtArea**.**setEditable**(false);**

recievedMessages**.**setEditable**(false);**

// Add ScrollPane to TextArea

JScrollPane scroll **=** **new** JScrollPane**(**recievedMessages**);**

scroll**.**setBounds**(**10**,** 10**,** 460**,** 280**);**

add**(**scroll**);**

// Set textarea to scroll when new text is received

DefaultCaret caret **=** **(**DefaultCaret**)** recievedMessages**.**getCaret**();**

caret**.**setUpdatePolicy**(**DefaultCaret**.**ALWAYS\_UPDATE**);**

// add actionisteners to button

sendBtn**.**addActionListener**(this);**

closeBtn**.**addActionListener**(this);**

clearBtn**.**addActionListener**(this);**

bkUpBtn**.**addActionListener**(this);**

// Display Date and Time

SimpleDateFormat myFormat **=** **new** SimpleDateFormat**(**"EEE, MMM d, ''yy"**);**

String dateString **=** myFormat**.**format**(new** Date**());**

dateLbl**.**setText**(**"" **+** dateString**);**

getContentPane**().**setBackground**(**Color**.**white**);**

setLocationRelativeTo**(null);** // center Jframe on screen

setVisible**(true);**

**}**

@Override

public void actionPerformed**(**ActionEvent e**)** **{**

Object target **=** e**.**getSource**();**

**try** **{**

readOnlineUsers**();**

**}** **catch** **(**IOException e1**)** **{**

e1**.**printStackTrace**();**

**}**

**if** **(**target **==** sendBtn**)** **{**

String msg **=** outGoingMsg**.**getText**();**

**if** **(**msg**.**length**()** **>** 0**)** **{**

**try** **{**

String timeStamp **=** sdf**.**format**(new** Date**());**

Details d1 **=** **new** Details**(**userName**,** timeStamp**,** msg**,** **true);**

ObjectMessage m1 **=** session**.**createObjectMessage**(**d1**);**

publisher**.**publish**(**m1**);**

outGoingMsg**.**setText**(**""**);**

outGoingMsg**.**requestFocusInWindow**();**

**}** **catch** **(**JMSException ex**)** **{**

ex**.**printStackTrace**();**

**}**

**}**

**}**

**if** **(**target **==** clearBtn**)** **{**

recievedMessages**.**setText**(**" "**);**

**}**

**if** **(**target **==** closeBtn**)** **{**

int confirm **=** JOptionPane**.**showConfirmDialog**(null,**

"Are you sure you wish to EXIT Chat?"**,** "Exit Chat"**,**

JOptionPane**.**YES\_NO\_OPTION**);**

**if** **(**confirm **==** JOptionPane**.**YES\_OPTION**)** **{**

**try** **{**

Details d1 **=** **new** Details**(**userName**,** " System Message "**,**

"has left the Chat"**,** **false);**

ObjectMessage m1 **=** session**.**createObjectMessage**(**d1**);**

publisher**.**publish**(**m1**);**

dispose**();**

System**.**exit**(**0**);**

**}** **catch** **(**Exception ex**)** **{**

System**.**out**.**println**(**"Exit Button Error " **+** ex**);**

**}**

**}**

**}**

**if** **(**target **==** bkUpBtn**)** **{**

int confirm **=** JOptionPane**.**showConfirmDialog**(null,**

"Are you sure you wish to Backup the chat text??"**,**

"Exit Chat"**,** JOptionPane**.**YES\_NO\_OPTION**);**

**if** **(**confirm **==** JOptionPane**.**YES\_OPTION**)** **{**

**try** **{**

FileWriter fWriter **=** **new** FileWriter**(**"D:\\myChatBackup.txt"**,**

**true);**

PrintWriter outputFile **=** **new** PrintWriter**(**fWriter**);**

recievedMessages**.**write**(**outputFile**);**

JOptionPane

**.**showMessageDialog**(null,**

"Chat Backup Successful! File can be found at

D:\\myChatBackup.txt"**);**

**}** **catch** **(**Exception ex**)** **{**

System**.**out**.**println**(**ex**);**

**}**

**}**

**}**

**}**

public void setMessageListener**()** **throws** JMSException **{** // add asynchronous message listener

subscriber**.**setMessageListener**(new** MessageListener**()** **{**

public void onMessage**(**Message message**)** **{**

**try** **{**

Details d1 **=** **null;**

**if** **(**message **instanceof** ObjectMessage**)** **{**

d1 **=** **(**Details**)** **((**ObjectMessage**)** message**).**getObject**();**

recievedMessages**.**append**(**"[" **+** d1**.**getTimeStamp**()** **+** "] "

**+** d1**.**getUserName**()** **+** " : " **+** d1**.**getMessage**()**

**+** "\n"**);**

**if(**d1**.**online **==** **false)**

**{**

removeUser**(**d1**.**getUserName**());**

**}**

**}** **else** **{**

System**.**out**.**println**(**"Error - not an Object message"**);**

**}**

**}** **catch** **(**Exception ex**)** **{**

ex**.**printStackTrace**();**

**}**

**}**

**});**

outGoingMsg**.**addKeyListener**(new** KeyAdapter**()** **{**

public void keyPressed**(**KeyEvent evt**)** **{**

**if** **(**evt**.**getKeyCode**()** **==** KeyEvent**.**VK\_ENTER**)** **{**

String msg **=** outGoingMsg**.**getText**();**

**if** **(**msg**.**length**()** **>** 0**)** **{**

**try** **{**

String timeStamp **=** sdf**.**format**(new** Date**());**

Details d1 **=** **new** Details**(**userName**,** timeStamp**,** msg**,**

**true);**

ObjectMessage m1 **=** session**.**createObjectMessage**(**d1**);**

publisher**.**publish**(**m1**);**

outGoingMsg**.**setText**(**""**);**

outGoingMsg**.**requestFocusInWindow**();**

**}** **catch** **(**JMSException ex**)** **{**

ex**.**printStackTrace**();**

**}**

**}**

**}**

**}**

**});**

**}**

public void readOnlineUsers**()** **throws** IOException **{**

**try**

**{**

File file **=** **new** File**(**"D:\\usersOnline.txt"**);**

FileReader fileReader**;**

fileReader **=** **new** FileReader**(**file**);**

BufferedReader bufferedReader **=** **new** BufferedReader**(**fileReader**);**

String line**;**

usersOnlineTxtArea**.**setText**(**""**);**

**while** **((**line **=** bufferedReader**.**readLine**())** **!=** **null)**

**{**

usersOnlineTxtArea**.**append**(**line**+**"\n"**);**

**}**

fileReader**.**close**();**

**}**

**catch** **(**FileNotFoundException e**)**

**{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**}**

public void removeUser**(**String user**)** **throws** IOException

**{**

**try**

**{**

BufferedReader in **=** **new** BufferedReader**(new** FileReader**(**

"D:\\usersOnline.txt"**));** // read in file containing current online Users

String str **=** **null;**

ArrayList**<**String**>** lines **=** **new** ArrayList**<**String**>();**

**while** **((**str **=** in**.**readLine**())** **!=** **null)** **{**

lines**.**add**(**str**);**

**}**

lines**.**remove**(**user**);** // remove userName from the ArrayList

PrintWriter pw **=** **new** PrintWriter**(new** FileOutputStream**(**

"D:\\usersOnline.txt"**));** // write ArrayList back to file

**for** **(**String line **:** lines**)**

pw**.**println**(**line**);**

pw**.**close**();**

**}**

**catch** **(**FileNotFoundException e**)**

**{**

e**.**printStackTrace**();**

**}**

**}**

public static void main**(**String**[]** args**)** **throws** Exception **{**

myChatGui myGui **=** **new** myChatGui**();**

myGui**.**outGoingMsg**.**requestFocusInWindow**();** // set focus to textfield

**}**

**}**

## LoginFrame.java

public class LoginFrame **extends** JFrame **implements** ActionListener **{**

private static final long serialVersionUID **=** 1L**;**

private JLabel jcomp1**;**

private JLabel jcomp2**;**

private JTextField userNameTf**;**

private JButton enterChatBtn**;**

public LoginFrame**()** **{**

**super(**"Login Screen"**);**

// construct components

jcomp1 **=** **new** JLabel**(**"Welcome to JMS Chat Application"**);**

jcomp2 **=** **new** JLabel**(**"Enter your Username : "**);**

userNameTf **=** **new** JTextField**(**5**);**

enterChatBtn **=** **new** JButton**(**"Enter Chat"**);**

// adjust size and set layout

setSize**(**510**,** 170**);**

setLayout**(null);**

// add components

add**(**jcomp1**);**

add**(**jcomp2**);**

add**(**userNameTf**);**

add**(**enterChatBtn**);**

// set component bounds (only needed by Absolute Positioning)

jcomp1**.**setBounds**(**155**,** 20**,** 195**,** 25**);**

jcomp2**.**setBounds**(**55**,** 80**,** 145**,** 25**);**

userNameTf**.**setBounds**(**190**,** 80**,** 125**,** 25**);**

enterChatBtn**.**setBounds**(**320**,** 80**,** 100**,** 25**);**

enterChatBtn**.**addActionListener**(this);** // add actionistener to button

getContentPane**().**setBackground**(**Color**.**WHITE**);**

setLocationRelativeTo**(null);** // center Jframe on screen

setVisible**(true);**

}

public void actionPerformed(ActionEvent e) {

Object target = e.getSource();

if (target == enterChatBtn) {

login();

}

}

public void login() {

try

{

this.setVisible(false);

myChatGui myGui = new myChatGui();

new Timer(1000, myGui).start(); // create timer

String userName = userNameTf.getText().trim();

myGui.userName = userNameTf.getText();

myGui.setTitle(userName + "'s Client");

Details d1 = new Details(myGui.userName, " System Message ",

"has joined the Chat ", true);

ObjectMessage m1 = myGui.session.createObjectMessage(d1);

myGui.publisher.publish(m1);

try

{

PrintWriter out = new PrintWriter(new BufferedWriter(new

FileWriter("D:\\usersOnline.txt", true)));

out.println(userName);

out.close();

}

catch (IOException ex)

{

System.out.println("Error Writting username to file");

}

}

catch (Exception e1)

{

e1.printStackTrace();

}

}

public static void main(String[] args) {

LoginFrame myLoginFrame = new LoginFrame();

}

}

## Details.java

**import** java**.**io**.**Serializable**;**

public class Details **implements** Serializable **{**

private static final long serialVersionUID **=** 1L**;**

String userName**;**

String timeStamp**;**

String message**;**

boolean online**;**

public Details**(**String userName**,** String timeStamp**,** String message**,** boolean online**)** **{**

**this.**userName **=** userName**;**

**this.**timeStamp **=** timeStamp**;**

**this.**message **=** message**;**

**this.**online **=** online**;**

**}**

public boolean getOnline**()** **{**

**return** online**;**

**}**

public void Online**(**boolean a**)** **{**

**this.**online **=** a**;**

**}**

public String getUserName**()** **{**

**return** userName**;**

**}**

public void setUserName**(**String userName**)** **{**

**this.**userName **=** userName**;**

**}**

public String getTimeStamp**()** **{**

**return** timeStamp**;**

**}**

public void setTimeStamp**(**String timeStamp**)** **{**

**this.**timeStamp **=** timeStamp**;**

**}**

public String getMessage**()** **{**

**return** message**;**

**}**

public void setMessage**(**String message**)** **{**

**this.**message **=** message**;**

**}**

**}**

# References

http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html

http://openjms.sourceforge.net/

http://openjms.sourceforge.net/usersguide/examples.html